НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Приладобудівний факультет

Кафедра автоматизації експериментальних досліджень

«На правах рукопису»
УДК 621.3.087.44

«До захисту допущено»
Завідувач кафедри
_____ Ю.М. Туз
«___»_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 152 «Метрологія та інформаційно-вимірювальна техніка»

на тему: «Організація систем інтернету речей на основі технологій .NET»

Виконав :
студент Vі курсу, групи ВА-81мп
Баштовий Максим Юрійович _____

Керівник:
доцент, к.т.н.
Богомазов С. А. _____

Консультант з “Розробки стартап-проектів”:
доцент, д.е.н.
Бояринова К.О. _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет
Кафедра автоматизації експериментальних досліджень

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою «Інформаційні вимірювальні технології та системи»

Спеціальність (спеціалізація) – 152 «Метрологія та інформаційно-вимірювальна техніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ю.М. Туз

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Баштовому Максиму Юрійовичу

1. Тема дисертації «Організація систем інтернету речей на основі технологій .NET», науковий керівник дисертації Богомазов Сергій Анатолійович доцент, к.т.н., затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації: 10.12.2018р.

3. Об’єкт дослідження: Система комплексної обробки подій

4. Предмет дослідження: Використання системи кмплексної обробки подій для обробки результатів вимірів

5. Перелік завдань, які потрібно розробити:

1. Огляд систем комплексної обробки подій.
2. Розробка апаратного забезпечення.
3. Моделювання подієво-керованих вимірювальних систем інтернету речей з використанням пакету gpss world.
4. Розробка стартап проекту «Організація систем інтернету речей на основі технологій .NET».
5. Висновки.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу : презентація, макет.

6. Орієнтовний перелік публікацій:

1. “ ОРГАНІЗАЦІЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ ТЕХНОЛОГІЙ .NET ” Баштовий М.Ю., Богомазов С.А.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Розробка стартап-проекту	Бояринова К.О. д.е.н., доцент		

9. Дата видачі завдання 09.09.2019

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Огляд існуючих рішень для створення системи	до 29.10.2018 р.	
2.	Розробка апаратного забезпечення системи	до 01.02.2019 р.	
3.	Розробка програмного забезпеченя мікроконтролера	до 20.06.2019 р.	
4.	Розробка імітаційної моделі системи	до 01.10.2019 р.	
5.	Розробка стартап-проекту.	до 22.11.2019 р.	
6.	Оформлення пояснювальної записки та ілюстративного матеріалу.	до 8.12.2019 р.	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації
доцент кафедри АЕД к.т.н, доцент _____ Богомазов С.А.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація на тему: «Організація систем інтернету речей на основі технологій .NET», 124 сторінок, 2 додатки.

Об’єкт дослідження: Системи комплексної обробки подій.

Предмет дослідження: Використання системи комплексної обробки подій для аналізу даних в системах інтернету речей.

Мета роботи: Розробка апаратно-программного забезпечення для системи інтернету речей з використанням комплексної обробки подій.

Методи дослідження та апаратура: Робота з джерелами інформації, пов’язаними з проблеми дослідження, експериментальні дослідження за допомогою програмного забезпечення на основі мов програмування C# та бібліотеки Nesper.

Результати роботи та їхня новизна: Розроблено програмне забезпечення для демонстрації обробки результатів вимірювань за допомогою комплексної обробки подій. Використання комплексної обробки подій дозволило обробляти дані в реальному часі.

Рекомендації щодо використання результатів роботи: Розроблене програмне забезпечення може бути використане в системах інтернету речей, в яких необхідно обробляти дані в реальному часі.

Ключові слова: ***комплексна обробка подій, інтернет речей.***

ABSTRACT

Master's thesis: " Organization of Internet of Things systems based on .NET technologies», 124 pages, 2 attachments".

Object of research: Systems of complex event processing.

Subject of research: Use of complex event processing system for data analysis in Internet of Things systems.

Objective: Development of hardware and software for the Internet of Things system using complex event processing.

Research methods and equipment: Work with sources of information related to the research problem, experimental research using software based on C # programming languages and the Nesper library.

Results of work and their novelty: Software was developed to demonstrate the processing of measurement results by means of complex event processing. The use of complex event processing allowed for real-time processing of data.

Recommendations on the use of work results: The developed software can be used on the Internet of Things, in which it is necessary to process data in real time.

Keywords*: complex event processing, internet of things.*

ЗМІСТ

ВСТУП

На сьогоднішній день, нас оточує широкий спектр електронних пристроїв, що використовуються для збору та генерації даних. Зазвичай ці пристрої пов'язані між собою і утворюють мережу, яка також включає інформаційні системи, а іноді й інші об'єкти. Це спрощений і мінімальний опис того, що називається Інтернетом речей (IoT). У мережі Інтернету речей є численні пристрої, кожен з яких генерує величезну кількість подій. Подія може бути визначена як усе, що відбувається, або вважається таким, що відбувається. Загалом, подія може бути вхідним електронним листом, фінансовою торгівлею, проколом шин, вказівкою на те, що світло увімкнене або вимкнено тощо. У мережах IoT подія може бути зчитуванням з датчика.

Інтеграція та аналіз створених даних є важливими. Таким чином, дані можуть надати цінну інформацію, так що можна зробити висновки і приймати рішення, засновані на даних. Можна відразу вказати на великі дані, оскільки ми використовуємо цей термін для позначення великих і складних сукупностей даних. Збільшення ємності для зберігання даних дозволило зберігати величезну кількість даних. У той же час, просування в обчислювальній потужності та штучному інтелекті сприяло більш ефективному аналізу даних. Однак дані, генеровані мережами IoT, є не лише кількісно великими, але й швидкими, оскільки вони створюються з величезними темпами. Тому виникає термін «Швидкі дані». Якщо дані спочатку зберігаються для того, щоб згодом їх обробляти, то втрачаються дорогоцінні можливості реагувати під час отримання даних. Отже, неможливість діяти в режимі реального часу змушує дані втрачати частину своєї цінності.

Для обробки подій в умовах швидких даних ми використовуємо методи комплексної обробки подій (CEP). Комплексна обробка подій - це обробка подій з декількох джерел з метою виявлення значущих подій та реагування на них якомога швидше. Простим прикладом може бути система виявлення пожежі будівлі. Ми можемо розмістити набір датчиків, які вимірюють

температуру по всій будівлі. Датчики направляють свої вимірювання в інформаційну систему, яка може зробити просте узгодження шаблонів і перевірити, чи маємо значення останніх вимірювань значно більші, ніж попередніх. Ми можемо використовувати і більш складні методи CEP. Якщо система виявить незвичне підвищення температури за короткий проміжок часу, вона генерує попередження. Основним завданням CEP є перетворення набору базових подій (наприклад, вимірювань) в одну або кілька складних подій, що несуть логічний смисловий зміст. Набір складається з різної кількості базових подій, що варіюється від кількох до тисяч. Крім того, сукупності складних подій можуть бути використані для отримання ще складніших подій.

Системи комплексної обробки подій повинні забезпечувати високу пропускну здатність подій та обробку подій «на льоту». Для виконання цих вимог системи CEP повинні бути надзвичайно масштабованими. Підвищення масштабованості може бути досягнуто розподіленими системами. У сучасну епоху хмарних обчислень хмарні системи стають вирішальною архітектурною частиною і є ідеальним пристосуванням для маніпулювання подіями, породженими мережами IoT. Однак не всі архітектури максимально використовують потенціал хмарних середовищ. Тому архітектура мікросервісів може бути кращим вибором, ніж традиційна монолітна архітектура для розміщення програм у хмарних середовищах.

1 ОСОБЛИВОСТІ ВИКОРИСТАННЯ КОМПЛЕКСНОЇ ОБРОБКИ ПОДІЙ В СИСТЕМАХ ІНТЕРНЕТУ РЕЧЕЙ

Керована подіями архітектура (EDA) - це архітектура, заснована на генеруванні, виявленні, використанні та реагуванні на події [1]. Комплексна обробка подій (CEP) - це підмножина EDA. Завдання CEP - фільтрувати, агрегувати та співставляти події низького рівня для генерування подій вищого рівня, тобто складних подій [2].

1.1 Необхідність використання CEP

Ми збираємо події (тобто необроблені дані) з пристроїв IoT, щоб перетворити їх на корисну інформацію. Один із способів зробити це - зберігати події в базі даних і згодом використовувати алгоритми для отримання результатів на основі даних. Однак такий підхід має істотний недолік. Якщо ми зберігаємо дані в базі даних, щоб пізніше їх обробити, ми втрачаємо можливість аналізувати дані, поки вони ще свіжі. У більшості випадків ми хочемо в режимі реального часу перетворити необроблені події з пристроїв IoT у змістовну інформацію. Ми знаємо, що події в мережах IoT створюються з великою швидкістю. Ось чому термін «Швидкі дані» є таким популярним у наш час. Хоча ми можемо використовувати базу даних для обробки подій у реальному часі, це не є доцільним. Запити до бази даних суттєво впливають на продуктивність обробки. Ця проблема стає ще більш очевидною в мережах IoT через велику кількість подій та високу швидкість створення подій. Тому такий підхід не відповідає потребам систем IoT.

Візьмемо для прикладу систему виявлення пожежі. Ми хочемо отримувати сповіщення про пожежу, як тільки вона починається. Час у таких ситуаціях є критичним, і від цього залежить застосування системи виявлення пожежі. У системах виявлення пожежі взагалі немає необхідності зберігати будь-які вихідні події. Якщо виявлено пожежу, відбувається складна подія. Складна подія включає всю релевантну інформацію про виявлену пожежу

(наприклад, місце пожежі) [3]. Тому використання бази даних у таких сценаріях є зайвим. Можна зробити висновок, що потрібен інший підхід.

CEP допомагає нам трансформувати вихідні події в відповідну інформацію «на льоту». IoT це не єдина область, де використовується CEP. CEP можна використовувати для аналізу: потоків транзакцій з виявлення шахрайства з кредитними картками, фінансових ринків для виявлення тенденцій, мережевого трафіку для виявлення вторгнень тощо [3]. У цьому розділі ми опишемо його еволюцію, порівнюючи його з попередниками, та розглянемо конкретну реалізацію - Nesper. Потім ми переглянемо список найпоширеніших конструкцій та операторів, які використовуються на мовах CEP. Використовуючи комбінації конструкцій та операторів, ми можемо аналізувати потік подій у режимі реального часу та отримувати різну інформацію.

1.2 Еволюція систем обробки потоків інформації

Необхідність обробляти величезну кількість даних у режимі реального часу спричинила розвиток систем для обробки потоку інформації (ОПІ).

Джерелами є сутності, які виробляють дані. Дані з джерел надходять у двигун ОПІ. Джерелами можуть бути різні сутності. У мережах IoT джерелами є пристрої IoT, які виробляють події. Джерелом може бути й інший двигун ОПІ. Коли потік даних надходить у двигун ОПІ, дані обробляються (наприклад, фільтруються, агрегуються, узгоджуються тощо) відповідно до визначених правил обробки. Правила обробки додають, керують та видаляють менеджери правил. Вихідні данні, що вироблені двигуном ОПІ відповідно до правил обробки, передається до приймачів. Серед них може бути двигун ОПІ, який здійснює подальшу обробку результатів.

Ми можемо бачити, що системи ОПІ повинні задовольняти деяким ключовим вимогам мереж IoT. По-перше, вони повинні вміти обробляти дані в режимі реального часу та вміти справлятися з величезними обсягами даних. По-

друге, вони повинні запропонувати мову для обробки даних, що дасть нам змогу ефективно визначати складні зв’язки між даними [3].

У наступних підрозділах проаналізовано еволюцію систем ОПІ: активні системи баз даних, системи управління потоками даних та найпопулярніші – системи комплексної обробки подій.

1.2.1 Активні системи баз даних

Ми отримуємо дані від традиційних систем управління базами даних (СУБД) у активному для людини та пасивному для бази даних способі, тобто пишемо запити та отримуємо дані. Активні системи баз даних доповнюють СУБД, намагаючись замість цього перемістити активну поведінку на рівні бази даних. Однак основою для цих систем є все ще постійне зберігання. Це означає, що вони все ще мають проблеми з величезною кількістю даних та великою кількістю правил обробки [3].

1.2.2 Системи управління потоками даних

Системи управління потоками даних (СУПД) намагаються подолати проблеми, які мають активні системи баз даних. Ми можемо встановлювати безперервні (стоячі) запити, які розгорнуті в системах баз даних і надавати результати, поки їх не видалимо. Тому ми отримуємо сповіщення, коли потік обробляється, без необхідності запускати багато запитів. Ми використовуємо ці системи в пасивним для людини та активним для бази даних способом. Однак СУПД не в змозі використовувати відносини послідовності та упорядкування, оскільки вони все ще є розширенням бази даних [3].

1.2.3 Системи комплексної обробки подій

На відміну від СУПД, що є розширенням бази даних, CEP - це розширення систем публікації-підписки на обмін повідомленнями. У системах публікації-підписки, коли ми підписуємось на тему, ми отримуємо поодинокі події, оскільки системи публікації-підписки не підтримують історію подій і не

виявляють зв’язків між ними. CEP розширює таку поведінку і дає змогу підписуватись на складні події Для того, щоб запропонувати нам підписку на складні події, у CEP відсутні основні недоліки систем СУПД. Системи СУПД не мають можливість задавати складні відносини між даними, та трудомісткі операції, такі як впорядкування, що дуже повільно виконується в базі даних. З іншого боку, системи CEP можуть робити складне узгодження шаблонів, фільтрацію, агрегацію, послідовність, упорядкування тощо. CEP дає нам набагато більше свободи у визначенні складних правил обробки. CEP намагається знайти зв’язок між подіями, що надходять, відповідно до визначених нами правил обробки. Якщо вона все-таки знайде зв’язок, то виникає складна подія[3].

CEP може також враховувати ієрархії подій. Складні події можна опрацьовувати далі, та генерувати події вищого рівня. Ми можемо мати стільки рівнів ієрархії, скільки потрібно [2].

1.3 Конструктори та оператори мов CEP

Практично кожен движок CEP має власну мову шаблонно-орієнтовану мову обробки подій. Шаблонно-орієнтовані мови задають умови та дії, які слід виконати, якщо умови дотримані. Умови зазвичай представлені шаблонами, які обробляють потоки за допомогою різних конструкцій та операторів. Правила перетворення (наприклад, виробляють два потоки з одного потоку) є специфічними для мов перетворення, які в основному використовуються в СУПД. Сучасні мови CEP (наприклад, мова обробки подій Esper) дозволяють нам також визначати правила перетворення [3]. Більшість мов не мають усіх мовних конструкцій та операторів, описаних у цьому розділі, а лише їх підмножину. Ми спробуємо охопити найпоширеніші конструкції та оператори мов CEP [3].

1.3.1 Одинарні оператори

Оператори з одним елементом є одними з найпростіших операторів у системах CEP. Як випливає з назви, вони обробляють окремі події самостійно, одна за одною. Існує два класи операторів з одним елементом:

- Оператори вибору - це класичні оператори фільтрації, які викидають події, які не відповідають заданим критеріям. Наприклад, в системі виявлення пожежі оператори відбору можуть фільтрувати події, які вказують на те, що температура вище певного порогу.
- Оператори перетворення. Ці оператори використовуються для трансформації подій. Деякі приклади - оператори проектування та перейменування. Оператори проекції витягують лише частину властивостей подій (наприклад, витягують лише місце зчитування датчиків). Оператори перейменування використовуються для зміни назви властивості події.

1.3.2 Логічні оператори

Логічні оператори використовуються для виявлення комбінацій подій. Ці оператори не враховують порядок подій. Є чотири групи логічних операторів:

- Кон'юнкція. Кон'юнкція подій E_1, E_2, …,E_n означає, що всі події E_1, E_2, …,E_n відбулися.
- Диз'юнкція. Диз'юнкція подій E_1, E_2, …,E_n означає, що хоча б одна з подій E_1, E_2, …,E_n відбулася.
- Повторення. Повторення події E зі ступенем (m; n) означає, що подія сталася між m і n разів.
- Заперечення. Заперечення події E означає, що подія E не відбулася.

Ми можемо комбінувати ці оператори (наприклад, використовувати як кон'юнкцію, так і диз'юнкцію) для формування більш складних шаблонів. Також логічні оператори часто використовуються в поєднанні з вікнами, які ми

опишемо в одному з наступних підрозділів. Вікна використовується для встановлення меж, щоб логічні оператори враховували лише підмножину потоку (наприклад, поєднання двох подій за останні три секунди).

1.3.3 Послідовності

Послідовності дуже схожі на логічні оператори. Різниця полягає в тому, що порядок подій важливий для послідовностей. Послідовність подій E_1, E_2, …,E_n означає, що всі події E_1, E_2, …,E_n відбулися у вказаному порядку.

1.3.4 Ітерації

В ітераціях ми вказуємо умову ітерації. Послідовності подій повинні відповідати заданій умові ітерації. Порядок подій важливий і для ітерацій. Ітерації не включаються до всіх мов обробки подій, оскільки вони можуть бути виражені за допомогою інших конструкцій обробки подій (наприклад, комбінації послідовностей та рекурсії).

1.3.5 Вікна

Вікна - це мовні конструкції, які дозволяють операторам обробляти лише частину потоку подій. Вони часто використовуються з різними операторами. Їх призначення залежить від того, чи є оператори блокуючими чи неблокуючими. Блокуючими є оператори, які повинні споживати весь потік подій, перш ніж вони генерують вихід (наприклад, заперечення та повтори із заданою більш високою межею). З іншого боку, неблокуючі оператори генерують результат у міру течії потоку подій, і вони можуть припинити генерувати вихід, як тільки виявлять необхідні події (наприклад, кон'юнктивні та диз'юнктивні оператори). Тому вікона є важливими для використання операторів блокування, оскільки вони задають кінцеву частину потоку, щоб оператори, що блокують, змогли отримувати вихід. Що стосується операторів, що не блокують, вікна роблять їх більш потужними, вказуючи, на яку частину потоку вони повинні дивитися.

Існує два типу вікон. Згідно з першим, вікна поділяються на логічні (тобто на основі часу, такі як події за останні три секунди) та фізичні (тобто на основі підрахунку, такі як три останні події).

Другий тип залежить від способу переміщення вікна. Відповідно до цієї класифікації вікна можуть бути:

- Фіксовані вікна. Межі нерухомих вікон не рухаються. Слід вказати і нижню, і верхню межу.
- Знакові вікна. Нижня межа віконних орієнтирів задається заздалегідь і не рухається. Верхня межа рухається, коли нові події входять у двигун CEP.
- Рухомі вікна. Вони є найбільш часто використовуваним типом вікон. Як і у знакових вікнах, верхня межа рухається у міру протікання подій. У рухомих вікнах нижня межа рухається з тим же темпом, що і верхня межа. Таким чином, стандартні рухомі вікна дозволяють нам переглядати останні три події або події за останні три секунди, як нижня межа рухається разом із верхньою межею. Розсувні вікна також мають певні зміни (наприклад, перекидання вікон).

1.3.6 Оператори управління потоком

Як випливає з назви, оператори управління потоками використовуються для управління потоками. Операторами управління потоком є:

- З'єднання. Оператор з'єднання – це блокуючий оператор, який використовується для об'єднання двох потоків (або частин двох потоків) в один.
- Оператори множин. Оператори множин використовують операції стандартного набору для маніпулювання потоками. Оператор об'єднання об'єднує багато потоків в один, який має унікальні події з усіх потоків. Оператор перетину об’єднує багато потоків в один, який має події, виявлені в кожному з потоків. Оператор «за винятком» приймає два потоки та видаляє події з першого потоку, які містяться у другому потоці.

Оператор видалення-дублікат займає один потік і видаляє дублюючі події.

- Дублікат. Копіює один і той же потік, так що копія надається кожному відмінному приймачу (наприклад, для подальшої обробки).
- Групування. Групує події потокового розділу відповідно до властивості події.
- Сортувати за. Сортує події потоку відповідно до властивості події.

1.3.7 Параметризація

Мова обробки подій повинна бути здатна підтримувати параметризацію в той чи інший спосіб. Нам часто потрібно обробляти події в одному потоці по відношенню до подій в іншому потоці. Наприклад, у системі виявлення пожежі ми можемо створити попередження про пожежу, якщо виявимо і високу температуру, і дим. Тому нам потрібно фільтрувати потік датчиків температури відповідно до потоку датчиків диму. Різні мови CEP вирішують це питання по-різному. Більшість сучасних мов пропонують параметризацію через оператор сполучення. Наприклад, для того, щоб виявити пожежу, ми можемо вказати, що температура повинна бути вище 45 С, а в останні 10 секунд повинен бути дим. Потім двигун CEP робить параметризацію всередині. У деяких старих мовах ми повинні використовувати оператор приєднання та об’єднувати потоки.

1.3.8 Агрегація

Оператори агрегації використовуються для агрегування подій у потоці. Типові приклади операцій агрегації - мінімум, максимум і середнє значення. Існує два типи агрегатів:

- Агрегації виявлення. Вони використовуються під час оцінювання умов правил обробки. Наприклад, ми можемо захотіти виявити події, що нижче середньої температури за останні десять показань датчика.

- Виробничі агрегації. Ці оператори використовуються для виробництва подій вищого рівня. Простим прикладом може бути генерація подій, що містять середнє значення за останні десять температурних показань.

Окрім типових агрегацій (мінімум, максимум тощо), деякі мови також дозволяють нам визначати спеціальні агрегації.

1.4 Бібліотека Nesper

NEsper – це бібліотека для обробки комплексних подій. NEsper пропонує мову Event Processing Language (EPL), що підтримує та розширює стандарт SQL та дозволяє записувати складні вирази над подіями.[4]

1.4.1 Вікна даних

Вікна даних – це один з засобів мови EPL, що дозволяє відсіювати події. Вікна вказують як довго зберігати відповідні події та за яких умов вони можуть бути відкинуті. Вікна можуть працювати на рівні окремих запитів, потоків та під запитів.

Вікно довжини вказує NEsper зберегти останні N подій з потоку. NEsper вводить всі події, що надходять в вікно довжини. Коли вікно довжини заповнюється, найстарша подія видаляється з вікна.

Рисунок 1.1 ілюструє, як вміст вікна довжини змінюється в міру надходження подій.

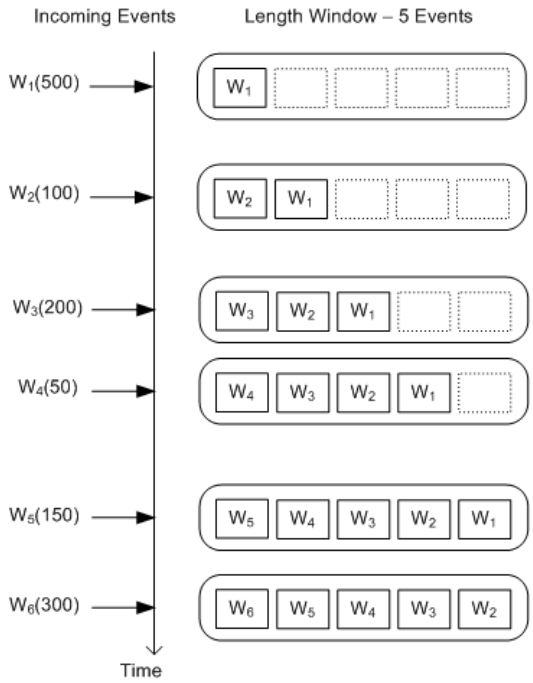

Рисунок 1.1 - Принцип роботи вікна довжини

Часове вікно - це рухоме вікно, що поширюється на заданий часовий інтервал у минуле на основі системного часу. Вікна часу дозволяють нам обмежувати кількість подій, які розглядаються за запитом, так само як і вікна довжини. Рисунок 1.2 ілюструє, як вміст вікна часу змінюється в міру надходження подій.

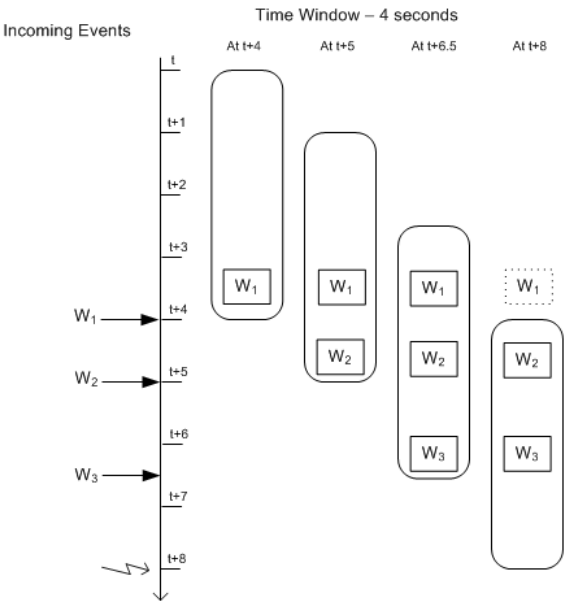

Рисунок 1.2 - Принцип роботи вікна часу

Пакетне вікно часу зберігає події та видаляє їх кожний заданий інтервал часу. Принцип роботи зображено на наступній діаграмі (рис. 1.3).

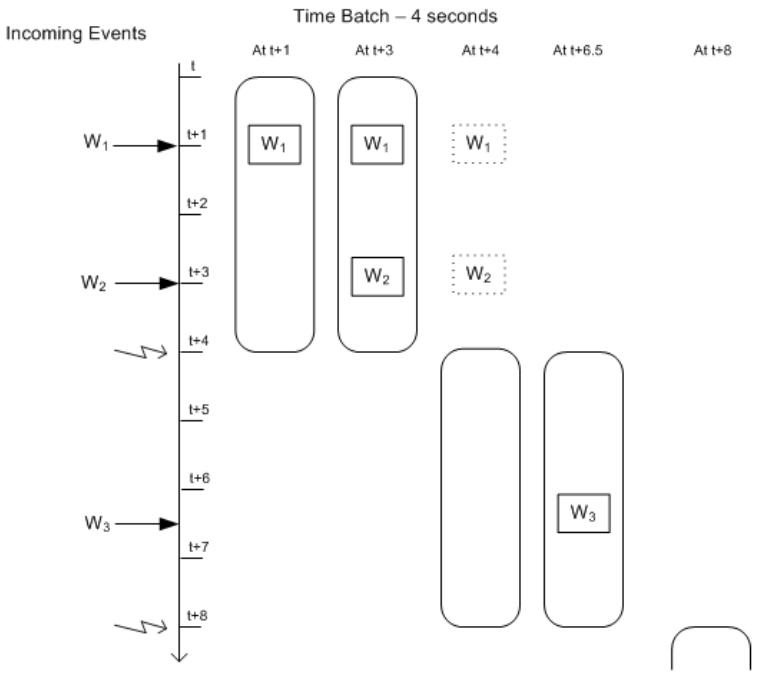

Рисунок 1.3 - Принцип роботи пакетного вікна часу

Іменовані вікна - це видимі в глобальному масштабі вікна даних, на які можна посилатися кількома операторами. Ви можете посилатися на імена вікон у операторах, які оголошують контекст без особливих міркувань, за винятком операцій on-action (останні повинні посилатися на той самий контекст, пов'язаний з іменованим вікном) тих самих подій у декількох місцях. Також є можливість створити іменне вікно і оголосити контекст для іменованого вікна. У цьому випадку NEsper керує окремими іменованими вікнами, по одному для кожного контекстного розділу. У цьому випадку застосовуються обмеження, які ми обговорюємо тут. Кожний on-merge, on-select, on-update та on-delete вираз мають бути об’явлені в одному і тому ж контексті.

1.4.2 Таблиці

Таблиці є глобально-видимими структурами даних, які містять рядки, організовані первинним ключем (ключами), і які можуть посилатися на декілька операторів. Є можливість посилатися на таблиці в операторах, які оголошують контекст, за винятком операторів on-action (останні повинні посилатися на той самий контекст, пов'язаний з таблицею). Таблиці оголошуються виразом create table.

Колонка таблиці може зберігати стан агрегації, що дозволяє спільно розміщувати дані з події та стан агрегації. Інші оператори можуть безпосередньо створювати та оновлювати загальний стан агрегації. Стан агрегації може включати в себе повний стан, такий як, наприклад, велику матрицю значень типу long для використання в наближенні ескізу Count-min. Після кожного типу таблиці можна написати ключове слово primary key, що позначає колонку як первинний ключ. Якщо декілька колонок позначені як первинні ключі, то значення цих колонок утворює складовий первинний ключ.

Ви також можете створити таблицю і оголосити контекст для таблиці. У цьому випадку NEsper фактично керує окремими таблицями, по одному для кожного контекстного розділу. У цьому випадку застосовуються обмеження, які ми обговорюємо тут. Коли ви оголошуєте контекст для таблиці, будь-які операції select, on-merge, on-select, on-update і on-delete, а також оператори, які під запиту до таблиці, повинні оголосити той самий контекст.

Речення insert into вставляє рядки в таблицю. Ваш запит повинен гарантувати, що імена та типи стовпців відповідають вказаним іменам стовпців і типам таблиці, у яку буде вставлено рядок.

1.4.3 Шаблони подій

Шаблони подій співпадають, коли відбувається подія або декілька подій, які відповідають визначенню шаблону. Шаблони також можуть бути засновані на часі.

Вирази шаблону складаються з атомів шаблону і операторів шаблону:

1. Атоми шаблонів є основними будівельними блоками шаблонів. Атоми - це вирази фільтрів, спостерігачі подій, заснованих на часі, і призначені для користувача модулі-спостерігачі, які спостерігають зовнішні події, які не перебувають під контролем NEsper.
2. Оператори шаблонів управляють життєвим циклом виразів і поєднують атоми логічно або тимчасово.

Є 4 типи шаблонних операторів:

1. Оператори, які керують повторенням підвирази шаблону: every, every-distinct, [num] and until
2. Логічні оператори: and, or, not
3. Часові оператори, які працюють за порядком подій: ->
4. Охоронці - це умови, які контролюють життєвий цикл підвиразів.

В основі зіставлення зі зразком лежить скінченний автомат, який здійснює перехід між станами на основі подій, що надходять, і на основі випереджаючого часу. Одна подія або час випередження може викликати реакцію в кількох частинах вашого активного стану шаблону. Шаблони знаходяться в деякому стані, так як Nesper підтримує стан шаблону. Шаблон може з'явитися де завгодно в реченні from оператора EPL, включаючи об'єднання і підзапити. Отже, шаблони можуть використовуватися в поєднанні з пропозицією where, group by, having, а також обмеженням rate виведення і insert into. Крім того, уявлення вікна даних може бути оголошено на шаблон. Вікно даних, оголошене в шаблоні, служить тільки для збереження відповідності шаблону. Вікно даних, оголошене в шаблоні, не обмежує, не скасовує і не видаляє проміжні відповідності шаблоном, коли відповідності шаблону залишають вікно даних.

Найпростіша форма фільтра - це фільтр для подій заданого типу без будь-яких умов для значень властивостей події. Цей фільтр відповідає будь-якій події цього типу незалежно від властивостей події.

Оператор every вказує, що підвираз шаблону має перезапускатися, коли підвираз, що обмежений ключовим словом every, оцінюється як true або false. Без оператора every підвираз шаблону зупиняється, коли підвираз шаблону оцінюється як true або false.

Подібний до оператора every, оператор every-distinct вказує, що підвираз шаблону має перезапускатися, коли підвираз, кваліфіковане ключовим словом every-distinct, оцінюється як true або false. Крім того, оператор every-distinct виключає повторювані результати, отримані з активного підвиразу відповідно до виразами різного значення.

1.4.4 Розпізнавання збігів

Синтаксис розпізнавання збігів представляє альтернативний спосіб завдання виявлення шаблону в порівнянні з мовою шаблонів EPL, описаним в попередньому розділі. Порівняння шаблонів розпізнавання збігів і EPL наведено нижче.

NEsper може застосовувати шаблони розпізнавання співпадінь в режимі реального часу після появи нових подій в потоці подій (також званих покроковими, потоковими або безперервними). NEsper також може зіставляти шаблони на вимогу через pull-API ітератора, якщо вказує іменоване вікно або вікно даних в потоці (таблиці не можуть використовуватися в реченні from з match-розпізнаванням). Механізм підтримує стан для часткових збігів із зразками, і тому шаблони розпізнавання співпадінь є конструкціями зі станом.

Ключове слово match_recognize запускає визначення визнання відповідності і відбувається відразу після пропозиції from в операторі вибору EPL. Далі йдуть дужки, які оточують визначення визнання відповідності.

Розділ на є необов'язковим і може використовуватися для вказівки, що події повинні бути розділені одним або кількома властивостями або виразами події. Якщо не існує жодного розділу, то всі рядки таблиці складаються з одного розділу. Регулярний вираз застосовується до подій в одному розділі, а не між розділами.

Пропозиція measures визначає стовпці, які містять вирази над змінних шаблонів. Вирази можуть посилатися на стовпці розділів, однозначні змінні, агрегати, а також індексовані властивості на змінні групи. Кожне вираження_вимірювання_вимірювання повинно супроводжуватися назвою як ключове слово і ім'я стовпця col_name.

Усі ключові слова, що збігаються, є необов'язковими і інструктують NEsper знаходити всі можливі збіги. За замовчуванням матчі ранжируються, і NEsper повертає єдиний матч, виконуючи алгоритм для усунення повторюваних збігів, як описано нижче. При вказівці всіх збігів матчі можуть перекриватися і можуть починатися з одного рядка.

Ключові слова після переходу не відповідають вимогам і слугують для визначення точки відновлення відповідності шаблону після виявлення відповідності. За замовчуванням поведінка відбувається після пропуску останнього рядка. Це означає, що після усунення повторюваних збігів логіка пропускає відновлення відповідності шаблону на наступній події після останньої події поточного збігу.

Компонент patterm використовується для вказівки регулярного виразу. Регулярний вираз будується з назв змінних і може використовувати такі квантори, як *, +,?, * ?, + ?, ??, {повторення} і | зміна (конкатенація позначається відсутністю будь-якого знака оператора між двома послідовними елементами у шаблоні).

За допомогою додаткового ключового слова interval, періоду часу або закінчення ви можете контролювати, як довго слід чекати, доки прибудуть інші події, які можуть бути частиною відповідної послідовності подій, перш ніж вказуватиметься на відповідність (або відповідності) (не застосовується до запиту на вимогу відповідності шаблонів).

Визначення необов'язкове і використовується для вказівки булевих умов, які визначають деякі або всі імена змінних, які оголошені в шаблоні. Ім'я змінної не вимагає визначення, і якщо немає визначення, типовим є предикат, який завжди є істинним. Таке ім'я змінної можна використовувати для відповідності будь-якому рядку.

1.4.5 Потоки даних

Потоки даних в NEsper EPL мають наступні цілі:

1. Підтримка програмування потоку даних і потокового програмування.
2. Декларативна інтеграція вхідних і вихідних адаптерів NEsper, які можуть бути надані NEsperIO або за допомогою програми.
3. Усунення необхідності використання шини подій для досягнення потоку даних тільки для видимості подій і типів подій для підвищення продуктивності.

Оператори потоку даних здійснюють зв'язок через потоки об'єктів події або подій, що переносяться. Основними об'єктами події є object, Dictionary, Object-array або DOM/XML. Загорнуті події представлені екземплярами EventBean, які пов'язують інформацію про тип з базовими об'єктами події.

NEsper пропонує ряд корисних вбудованих операторів, які можна об'єднати в графіку для програмування потоку даних. Крім того, NEsperIO пропонує готові оператори, які діють як джерела або поглиначі подій. Програма може легко створювати та використовувати власні оператори потоку даних.

Використовуючи потоки даних, програма може надавати події операторам потоку даних безпосередньо без використання шини подій двигуна. Невикористання шини подій (як представлено EPRuntime.sendEvent) дозволяє досягти збільшення продуктивності, оскільки двигун не потребує відповідності подіям до операторів, і двигун не потребує перенесення базових об'єктів подій у екземпляри EventBean.

Ваша програма оголошує потік даних за допомогою create dataflow dataflow-name. Оголошення потоку даних призводить до того, що компілятор EPL перевіряє синтаксис і деякі аспекти графіка потоків даних операторів. Оголошення потоку даних насправді не створює екземпляр або виконує потік даних. Вирішення типів подій і операторів екземплярів (за потребою) відбувається під час реалізації потоку даних.

Після того як ваша програма оголосила потік даних, вона може створити екземпляр потоку даних і виконати його. Потік даних може бути екземпляром стільки разів, скільки потрібно, і кожен екземпляр потоку даних може бути виконаний тільки один раз.

1.5 Розробка програмного забезпечення систем Інтернету речей з використанням .NET-бібліотеки NEsper

У платформі IoT є різноманітне обладнання, включаючи датчики температури та вологості, електромагнітного спектру, електронні камери та ін. Одразу після введення системи в експлуатацію ці фізичні пристрої будуть

постійно подавати зібрані дані в систему. Тоді система виведе небезпечні або цільові події з масиву даних у режимі реального часу за визначеним користувачем правилом та транспортує цільові події до верхніх додатків. В результаті ця платформа може реалізувати повний діапазон моніторингу в цільовій області. Більше того, вона може виявляти небезпечні події в режимі реального часу.

Основні функціональні вимоги платформи:

- Здійснення моніторингу у реальному часі за бізнес-сценарієм та отримання подій високого рівня з масових подій низького рівня. Крім того, платформа повинна бути гнучкою, щоб відповідати різноманітним складним бізнес-сценаріям та забезпечувати зручний, автоматизований інтерфейс динамічного управління подіями.
- Підтримання розподіленого механізму передачі даних.

На основі різних функцій ця платформа поділена на 3 окремі частини: модуль доступу до ресурсів, модуль видавець-підписник та модуль обробки комплексних подій.

- Модуль доступу до ресурсів відповідає за роботу фізичного пристрою відповідно до власних протоколів, структурування необроблених даних та публікацію структурованих даних у модуль видавець-підписник.
- Модуль Видавець-підписник базується на механізмі розподіленої диспетчеризації подій і відповідає за обмін даними в реальному часі між кожним модулем.
- Комплексний модуль обробки подій, на який фокусується ця стаття, відповідає за виведення події високого рівня з високошвидкісного потоку подій у режимі реального часу та повідомлення потоку цільових подій також до модуля видавець-підписник.

1.5.1 Модуль видавець-підписник

Архітектура програмного забезпечення "публікація - підписка" - це схема обміну повідомленнями, де відправники повідомлень, які називаються видавцями, не запрограмовані відправляти повідомлення конкретним

отримувачам, які називаються підписниками, замість цього опубліковані повідомлення розбиваються на категорії за класами, без знання про те, яким підписникам вони мають бути прийняті і чи взагалі будуть такі підписники. Так само, підписники виявляють зацікавленість в певних класах повідомлень і приймають ті повідомлення, які їх цікавлять, без знання того, які видавці їх публікують.

У моделі видавець-підписник, підписники зазвичай отримують лише підмножину загальної кількості опублікованих повідомлень. Процес вибору повідомлень для прийому та обробки називається фільтруванням. Існує дві поширені форми фільтрації: тематична та контентна.

У тематичній системі повідомлення публікуються на "теми" або логічні канали. Підписники в тематичній системі отримуватимуть усі повідомлення, опубліковані на теми, на які вони підписані, а всі підписники теми отримають однакові повідомлення.

У контентній системі повідомлення надсилаються абоненту лише у тому випадку, якщо атрибути чи вміст цих повідомлень відповідають обмеженням, визначеним абонентом. Абонент несе відповідальність за класифікацію повідомлень.

У платформі IoT видавці, як і датчики, публікують свої дані, а датчики такого ж типу зазвичай мають однакову форму даних. А абоненти, як-от модуль комплексної обробки подій, зосереджуються на даних деяких певних датчиків, щоб вони могли оброблятись далі. Тому шаблон на основі тем більше підходить для платформи IoT.

Для демонстрації можливостей пакету NEsper розроблена програма, що емулює систему пожежної сигналізації.

Спершу опишемо класи, що описують результати виміру температури та кімнату в якій розташований датчик температури.

```
public class TemperatureEvent
{
```

```
 public decimal Value { get; set; }
 public int SensorId { get; set; }
}
```

Клас, що описує температуру має дві властивості: значення температури та ідентифікатор датчику, з якого було отримано дане значення температури.

```
public class Room
{
 private List<int> _sensorIds = new List<int>();
 public int Number { get; set; }
 public int Floor { get; set; }
 public IEnumerable<int> SensorIds => _sensorIds;
 public void AddSensorId(int id) => _sensorIds.Add(id);
}
```

Клас, що описує кімнату містить номер кімнати, поверх, на якому розташована кімната та список датчиків температури, що розташовані у кімнаті.

Далі на основі класу кімнати створюємо десяти поверхову будівлю на кожному поверсі якої по десять кімнат, а в кожній кімнаті по п’ять датчиків температури.

```
var sensorIdConter = 1;
var rooms = new List<Room>();

for (int floor = 0; floor < 10; floor++)
{
 for (int number = 0; number < 10; number++)
 {
      var room = new Room
      {
           Id = 10 * floor + number + 1,
           Floor = floor + 1,
           Number = 10 * floor + number + 1
      };
      for (int sensor = 0; sensor < 5; sensor++)
```

```
            {
                room.AddSensorId(sensorIdConter++);
            }

            rooms.Add(room);
        }
    }
```

Далі створюємо двигун системи комплексної обробки подій та реєструємо типи подій.

```
    var engine = EPServiceProviderManager.GetDefaultProvider();
    engine.EPAdministrator.Configuration.AddEventType<TemperatureE
vent>();
    engine.EPAdministrator.Configuration.AddEventType<Room>();
    engine.EPAdministrator.Configuration.AddEventType("RoomToSenso
r", new Dictionary<string, object>() {
                    ["SensorId"] = "int",
                    ["RoomId"] = "int"
                });
```

Після реєстрації типів подій створюємо два вирази: перший для штатної роботи системи, що повертає ковзаюче середнє температури в кімнаті за останні 30 секунд, а другий для виявлення пожеж.

```
    var avaregeTemperatureStatemant = $@"
        select
            avg(Value) as roomTemperature,
            R.Number as roomNumber,
            R.Floor as roomFloor
        from TemperatureEvent#time(30 sec)
            join          RoomToSensor#length({rooms.Sum(x          =>
x.SensorIds.Count())})   as   RTS   on   TemperatureEvent.SensorId   =
RTS.SensorId
            join Room#length({rooms.Count}) as R on RTS.RoomId = R.Id
        Group by R.Number, R.Floor
```

```
     Order by R.Number";
    var                              stat                              =
engine.EPAdministrator.CreateEPL(avaregeTemperatureStatemant);
    var fireStatemant = $@"
     select
          R.Number as roomNumber,
          R.Floor as roomFloor
     from TemperatureEvent#time(30 sec)
          join          RoomToSensor#length({rooms.Sum(x          =>
x.SensorIds.Count())})   as   RTS   on   TemperatureEvent.SensorId   =
RTS.SensorId
          join Room#length({rooms.Count}) as R on RTS.RoomId = R.Id
     Group by R.Number
     Having avg(Value) > 49.95";
    var                             statFire                             =
engine.EPAdministrator.CreateEPL(fireStatemant);
```

Після створення виразів посилаємо в двигун події типу Room та RoomToSensor.

```
    foreach (var room in rooms)
    {
     engine.EPRuntime.SendEvent(room);
     foreach (var sensorId in room.SensorIds)
     {
          engine.EPRuntime.SendEvent(new Dictionary<string, object>
{    ["SensorId"]    =    sensorId,    ["RoomId"]    =    room.Id    },
"RoomToSensor");
     }
    }
```

Далі створюємо об’єкт блокування паралельний потік, що додає в двигун результати вимірів температури.

```
    var locker = new object();
```

```
var sender = new Thread(() => {
 var rand = new Random((int)DateTime.Now.Ticks);
 while (true)
 {
      lock (locker)
      {
           engine.EPRuntime.SendEvent(new TemperatureEvent()
           {
                SensorId = rand.Next(sensorIdConter),
                Time = DateTime.Now,
                Value = (decimal)(50 * rand.NextDouble())
           });
      }
      Thread.Sleep(1);
 }
});
sender.Start();
```

Далі додаємо обробник події отримання нового значення з виразу, що виявляє пожежі.

```
var alarmed = false;

statFire.AddEventHandlerWithReplay((s, e) => {
 if (e.NewEvents is null)
 {
      return;
 }
 Console.Clear();
 Console.ForegroundColor = ConsoleColor.Red;
 alarmed = true;

 foreach (var res in e.NewEvents)
 {
      Console.WriteLine("FIRE in room {0} on {1} floor",
res.Get("roomNumber"), res.Get("roomFloor"));
```

```
     }
    });
```

Після додання обробника подій виводимо значення середньої температури по кімнатах до тих пір поки не отримаємо сигнал про пожежу.

```
    while (!alarmed)
    {
     Console.Clear();
     lock (locker)
     {
          foreach (var res in stat)
          {
               Console.WriteLine("avarage temperature {0:N2} in room {1} on {2} floor", res.Get("roomTemperature"), res.Get("roomNumber"), res.Get("roomFloor"));
          }
     }
     Thread.Sleep(1000);
    }
```

Приклад виконання програми представлений на рисунку 1.4.

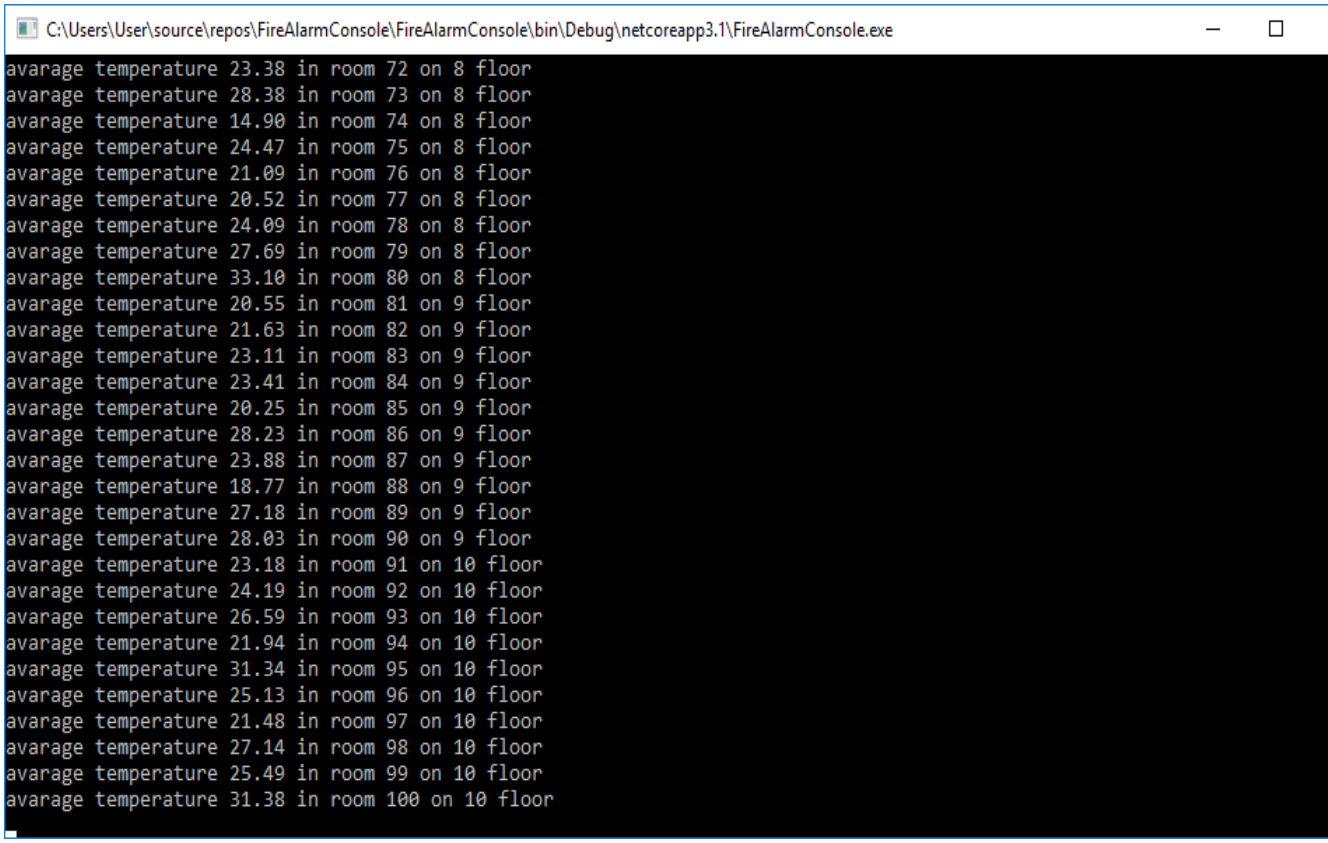

Рисунок 1.4 - Приклад виконання програми

Висновки

Модуль комплексної обробки подій на платформі IOT реалізує основне завдання - отримання даних високого рівня з високошвидкісних даних низького рівня в реальному часі, підтримку налаштованих правил EPL та управління роботою для користувачів та створення додатків, гнучких до будь-якого бізнес-сценарію. У майбутньому робота модулю CEP повинена забезпечити більш прозорий графічний інтерфейс для конфігурації правил EPL, щоб знизити складні для звичайних клієнтів конфігурації правил та покращити продуктивність при низькій затримці та високих паралельних обчисленнях.

2 РОЗРОБКА ПІДСИСТЕМИ ЗБОРУ ДАНИХ НА БАЗІ ПЛАТФОРМИ ІНТЕРНЕТУ РЕЧЕЙ ELECTRIC IMP

2.1 Особливості роботи з модулем Electric Imp

Платформа Electric Imp це надійне рішення, яке збирає в собі інтегрування з апаратним програмним забезпеченням, програмним забезпеченням, операційною системою, APIs, хмарними середовищами та безпекою[5]. Дана платформа може покращити пристрої в системі ІтР з більшою функціональністю, ефективністю та продуктивністю (рисунок 2.1).

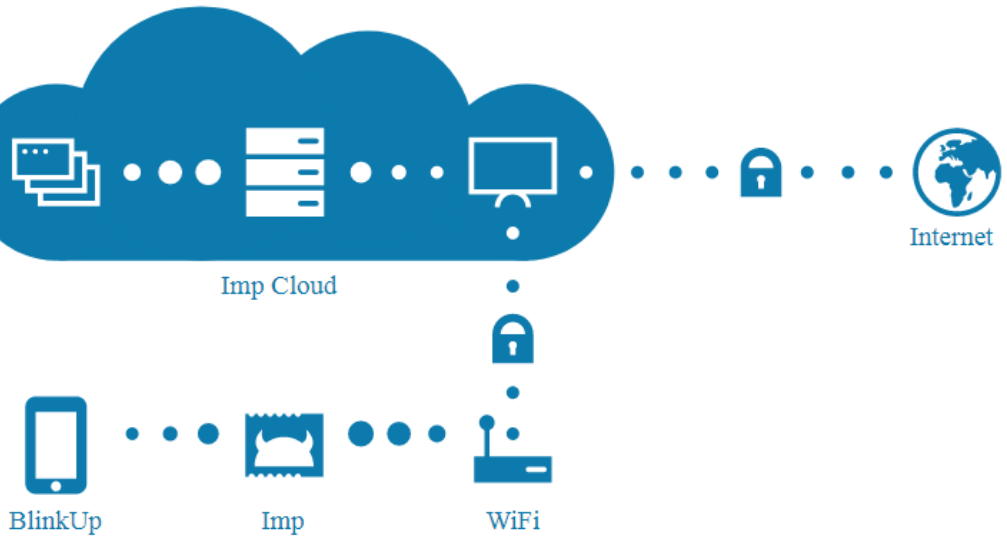

Рисунок 2.1 - Платформа Electric Imp

До особливостей платформи можна віднести:

- imp-Authorized Hardware;
- impOS;
- impSecure;
- BlinkUp;
- impCloud;
- imp APIs;
- impCentral;
- impCentral Production Tools;

- impCentral API;
- imp Code Libraries;
- impFactory.

imp-Authorized Hardware: платформа Electric Imp починається з імпу, потужного модуля, який інтегрує Wi-Fi та обчислює потужність, яка виступає в якості шлюзу для підключення вашого пристрою або сервісу до Інтернету. Імп апаратура доступна в декількох формах: модуль з вбудованою антеною або без неї, та як єдиний чіп для великомасштабних програм.

impOS: операційна система для функцій імпу, impOS надає захищені послуги з підключення до Інтернету для вашого продукту, і це дозволяє зосередитися на створенні коду, щоб забезпечити життя вашому продукту.

impSecure: функції захисту від Electric Impact розроблені та вбудовані в кожний фізичний та віртуальний аспект платформи Electric Imp. Вони постійно підтримуються та оновлюються.

BlinkUp: запатентована технологія BlinkUp являється унікальним способом скоротити підготовку та технічну експертизу, необхідну для надійного користування, надання та розгортання підключених пристроїв на платформі Electric Imp. Він використовує світлові схеми, щоб можна було безпечно з'єднуватися з хмарою за лічені секунди за допомогою стандартних смартфонів або планшетів.

impCloud: дозволяє запускати "agents", тобто код сервера, унікальний для кожного пристрою для забезпечення обробки HTTP IO та хмарної обробки всіх продуктів, а також легко підключити продукти до будь-яких пристроїв з Інтернетом. Агенти можуть бути центральним хабом для продуктів, додатків, сторонніх служб і навіть власних серверів.

imp APIs: використовуйте розширені можливості imp APIs для покращення взаємодії та побудови свого бізнесу шляхом розробки додатків, на прикладі обміну повідомленнями, моніторингу та багато іншого.

impCentral: середовище дозволяє реалізовувати та підтримувати програмне забезпечення, гарантуючи, що у користувачів завжди будуть найновіші функції.

impCentral API: дозволяє інтегрування з уже існуючими рішеннями та пристроями.

imp Code Libraries: існують готові рішення для підвищення продуктивності розробки програмного забезпечення.

Комплект Electric Imp impExplorer це ідеальна основа для створення автономних пристроїв. Він включає не тільки тріо температури та вологості, руху та датчиків тиску, але також індикатор RGB для видимого зворотного зв'язку. Крім того, він містить grove headers для розширення. Два з цих роз'ємів призначені для периферійних пристроїв I²C, інші для аналогових та/або цифрових пристроїв. На рисунку 2.2 показано вигляд комплекту impExplorer.

Рисунок 2.2 - Комплект impExplorer

До складу комплекту impExplorer входить стандартний міні-USB кабель, який подається від адаптера змінного струму або відповідного USB-порту на іншому пристрої, такому як комп'ютер. USB-порт використовується лише для живлення - передати дані таким чином неможливо[5].

Нижня частина комплекту impExplorer оснащена тримачем для трьох батарей типу AA. Комплект impExplorer завжди використовуватиме USB живлення, якщо воно буде присутнє, незважаючи на наявні батареї.

Для підтримки високопродуктивних (5 В) периферійних пристроїв, таких як індикатор RGB та використання портів Grove Analog / Digital, на початку коду пристрою слід включити наступний рядок:

```
hardware.pin1.configure(DIGITAL_OUT, 1);
```

Комплект включає в себе чотири заголовки, сумісні із системою Grove (рисунок 2.3), кожна з яких підключається до чотирьохпроводного кабелю з двома кабелями для даних (жовтий та білий), 3V3 (червоний) та GND (чорний).

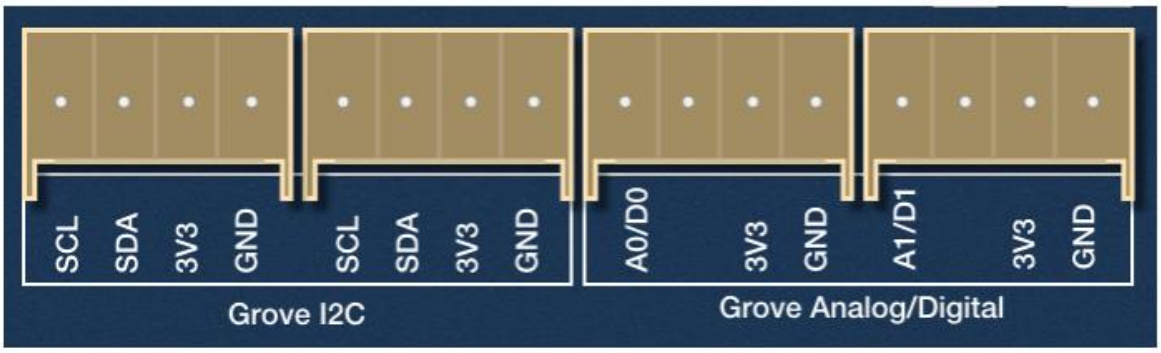

Рисунок 2.3 - Система Grove комплекту impExplorer

Провідники даних можуть бути використані для керування різноманітними пристроями, сумісними з Grove системою, хоча тільки ті, які працюють над I2C, підтримуються двома заголовками Grove системи I2C комплекту impExplorer, обидва з яких підключаються до шини hardware.i2c89. Жовтий провід забезпечує лінію clock; білий провід лінії передачі даних. Підтримуються лише пристрої 3V3 I2C.

Інші два заголовки призначені для цифрових та аналогових периферійних пристроїв, вони підключені до hardware001.pin2 та hardware.pin5 (A0/D0 і

A1/D1, відповідно). Ці піни з'єднуються лише з первинними проводами даних (жовтий) - вторинні проводи заголовків (білі) відключаються.

Imp001 - повний бездротовий мережевий вузол у формфакторі картки. Він працює разом із службою imp, що дозволяє легко підключати будь-який пристрій до Інтернету. Однією з переваг використання Wi-Fi всередині знімної картки користувача є те що всі схвалення бездротового регулювання відбуваються на рівні карти. Це полегшує необхідність сертифікації бездротового регулятора на рівні продукту.

Характеристики Imp001:

- 802.11 b/g/n WiFi
- 32бітний процесор Cortex M3
- вбудований двокольоровий LED для відображення статусу
- вбудований фототранзистор для BlinkUp технології
- SPI (2 канали), UART (3 канали), I2C (2 канали)
- GPIO, PWM, Analog IO
- енергозберігаючий режим при 6 мкА

Піни в карті 001 мають наступне призначення: VSS - земля, VDD - живлення, PIN1, 2, 5, 7, 8, 9 - ввід/вивід, ID - підключення до чипу Atmel ATSHA Id. На рисунку 2.4 зображено карта Imp001 з вказаними пінами.

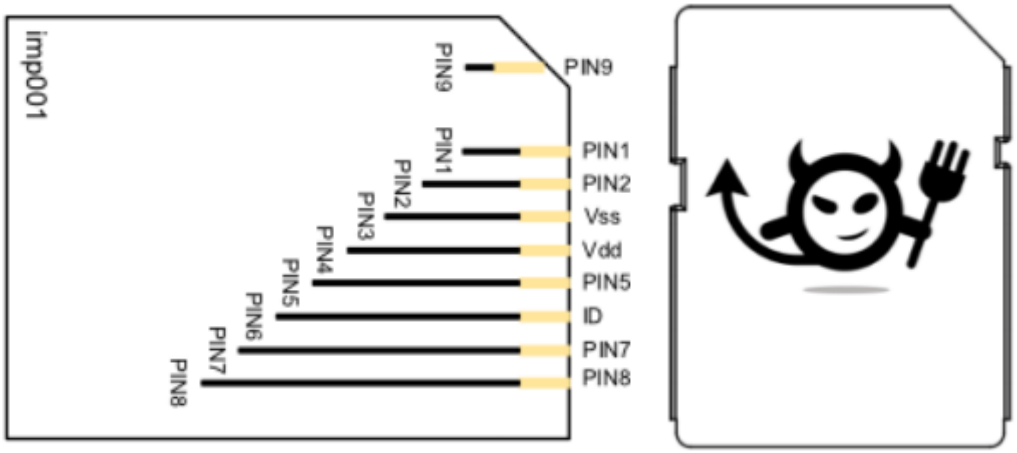

Рисунок 2.4 - Карта Imp001

Крім того, що піни виступають в ролі GPIO, кожен пін на Imp001 може бути налаштований на одну з декількох спеціалізованих функцій. Хоча піни можуть мати тільки одну функцію одночасно, вони можуть бути переналаштовані під час виконання, щоб змінити функцію як необхідно. Наприклад, спочатку пін може бути налаштований як ЦАП, а потім переналаштований як АЦП. Крім того, не всі піни в апаратній функції повинні бути використані. Наприклад, піни 8 і 9 можуть бути використані як UART та піни 1 і 2 можуть бути використані як I2C[5]. Можливості використання пінів зображені на рисунку 2.5.

Pin	GPIO	UART	I2C	SPI	DAC	ADC	PWM	Pulse Count	Wake
PIN1	Yes	U1-CTS, U3-TX	I1-SCL	SPI1-SCLK	Yes	Yes	Yes	Yes	Yes
PIN2	Yes	U1-RTS, U3-RX	I1-SDA	SPI2-MISO		Yes	Yes		
PIN5	Yes	U2-TX		SPI2-SCLK	Yes	Yes	Yes		
PIN7	Yes	U2-RX		SPI2-MOSI		Yes	Yes		
PIN8	Yes	U1-TX	I2-SCL	SPI1-MOSI		Yes	Yes		
PIN9	Yes	U1-RX	I2-SDA	SPI1-MSO		Yes	Yes		

Рисунок 2.5 - Можливості пінів Imp001

Imp001 можна розбудити з режиму низького енергоспоживання на PIN1. Якщо це імпульсний сигнал, то мінімальна ширина імпульсу 20 мс.

Для більшості додатків Imp бере на себе відповідальність за контролем вбудованого Wi-Fi чипу та за повторне підключення в разі втрати зв'язку за будь-яких причин. Однак, для спеціальних додатків, програмне забезпечення на мові Squirrel може ввімкнути або вимкнути Wi-Fi напряму через код.

impOS працює в двох режимах - SUSPEND_ON_ERROR та RETURN_ON_ERROR, SUSPEND_ON_ERROR - режим з автоматичним повторним підключенням та має більш доступну назву як "простий" режим (рисунок 2.6).

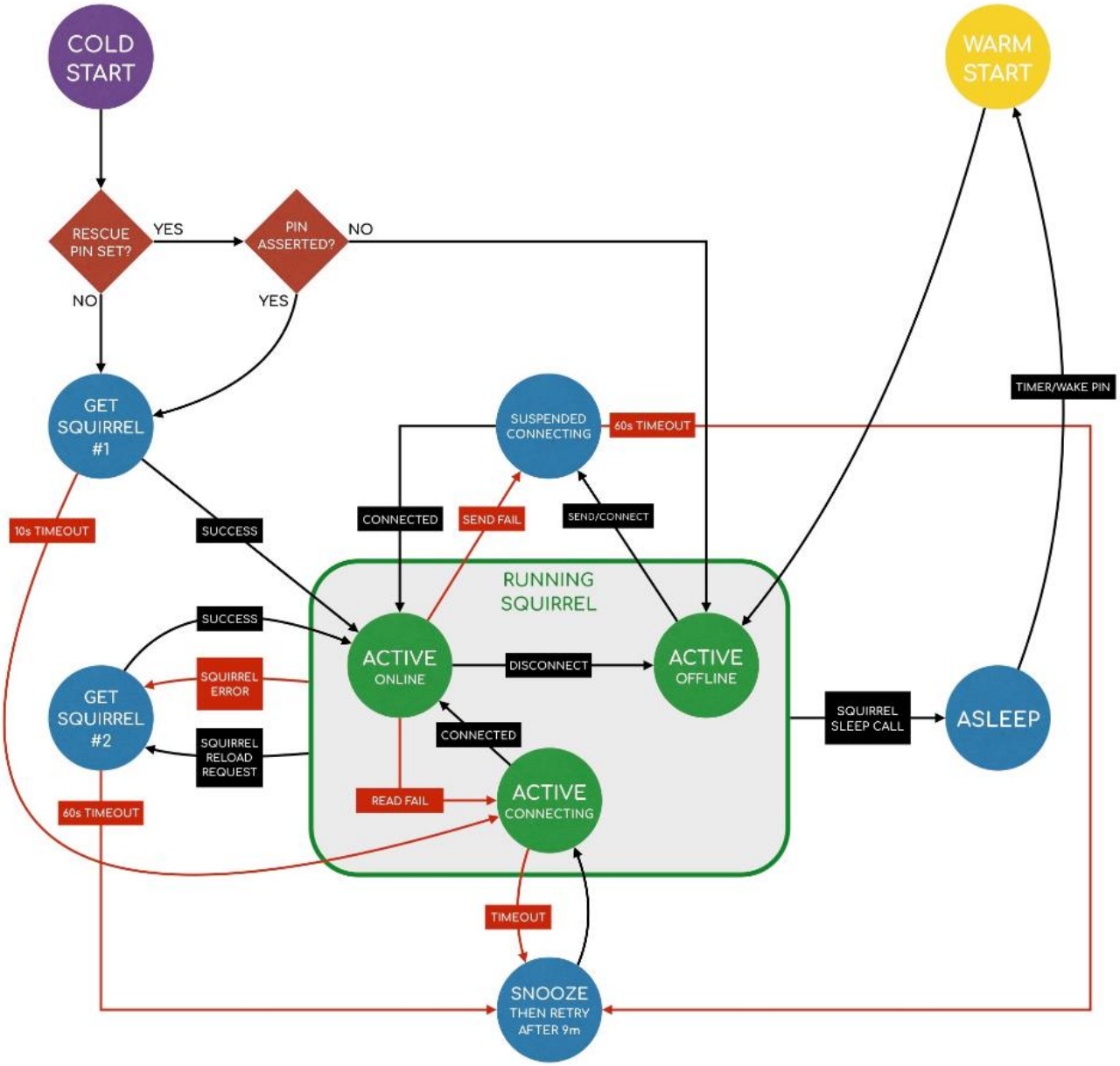

Рисунок 2.6 - Діаграма "простого" режиму

RETURN_ON_ERROR - режим з ручним повторним підключенням та має більш доступну назву як "експертний" режим (рисунок 2.7).

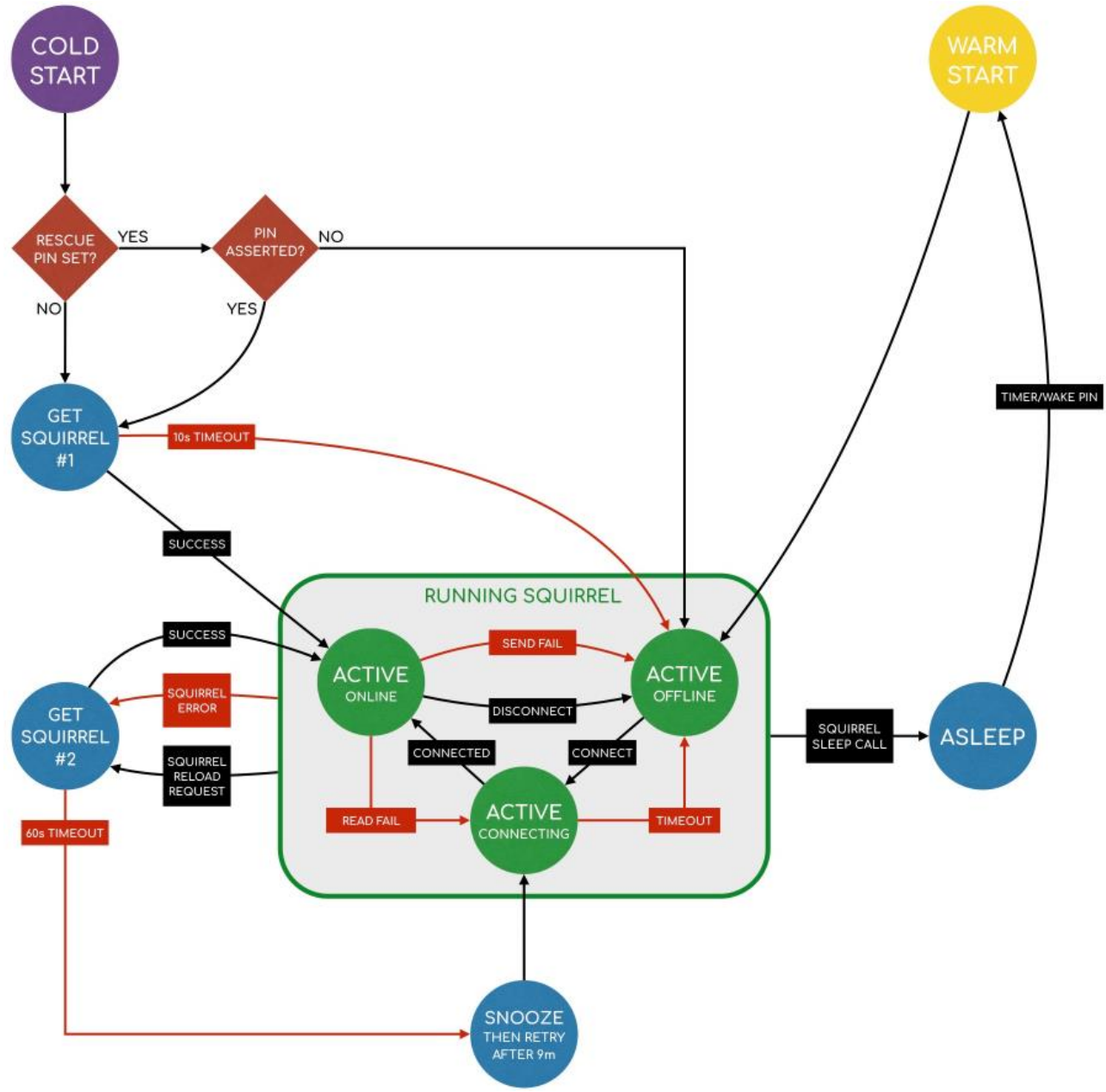

Рисунок 2.7 - Діаграма "експертного" режиму

COLD START: всі імпи починають тут коли живлення вперше подалося.

RESCUE PIN: перевірка чи був встановлений пін.

GET SQUIRREL #1: impOS намагається приєднатися до сервера, вмикаючи Wi-Fi чіп якщо необхідно.

ACTIVE ONLINE: імп приєднався до сервера та Squirrel код відпрацьовує.

ACTIVE CONNECTING: імп не приєднався до сервера та impOS намагається зробити повторне підключення.

ACTIVE OFFLINE: якщо імп використовує Wi-Fi, він вимикається для збереження енергії, Squirrel код продовжує відпрацьовувати.

GET SQUIRREL #2: імп не має ніякого Squirrel коду, тому тут генерується помилка.

SUSPREND CONNECTING:

SNOOZE: сплячий режим, Wi-Fi вимикається.

ASLEEP: сплячий режим, Wi-Fi вимикається.

WARM START: оновлення коду Squirrel, дані зберігаються в таблиці nv.

2.2 Налаштування проекту в середовищі impCentral

Electric Imp забезпечує два impClouds: один розміщений на Amazon Web Services (AWS), а інший розміщений на Microsoft Azure. Ви можете створити обліковий запис в impCloud, однак розробки пристроїв пов'язані з конкретними імпортуваннями даних, тому вам потрібно створити обліковий запис на impCloud, який підходить до вашого пристрою. Наприклад, якщо ви створюєте обліковий запис на Microsoft Azure, вам потрібно буде використовувати апаратуру, позначену як Microsoft Azure. Обліковий запис AWS не може бачити пристрої, пов'язані з Azure impCloud, і навпаки.

Для отримання доступу до середовища impCentral слід створити Electric Imp аккаунт. Необхідно перейти на веб-сторінку https://impcentral.electricimp.com/login (рисунок 2.9).

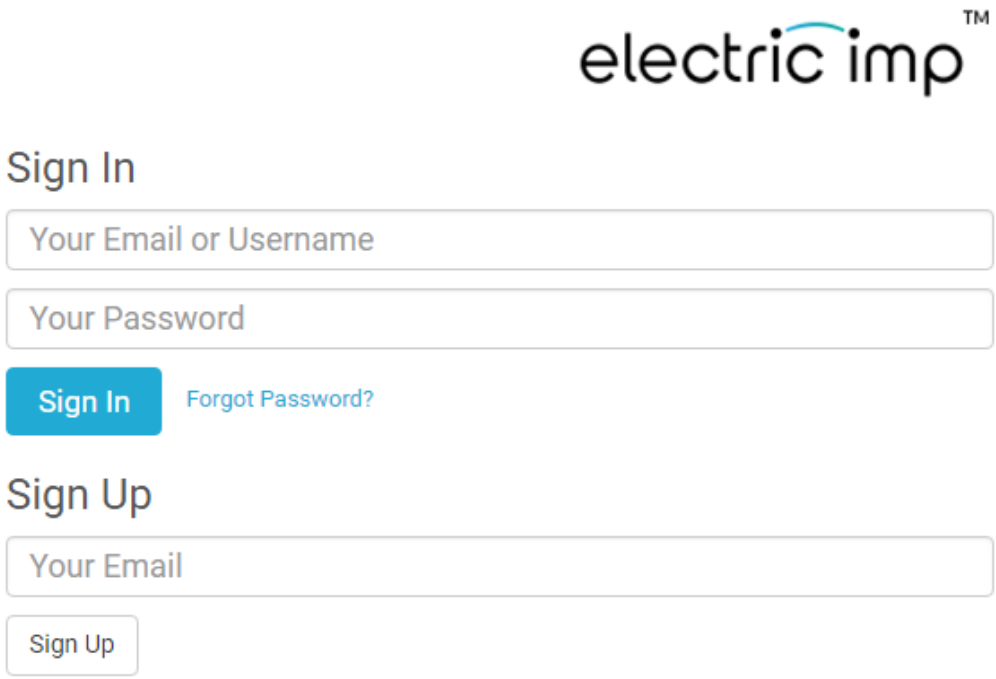

Рисунок 2.9 - Сторінка реєстрації / логіну AWS impCentral

Для реєстрації необхідно ввести електронну адресу та натиснути кнопку "Sign Up", за досить короткий проміжок часу буде надіслано лист на вказану електронну адресу (рисунок 2.10).

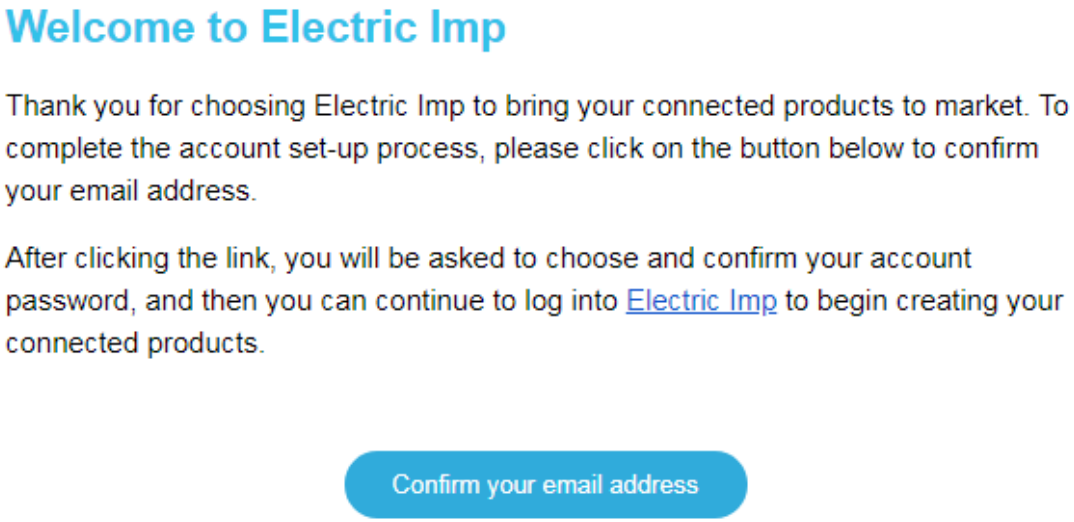

Рисунок 2.10 - Лист підтвердження

Далі треба перейти на вказану електронну адресу на натиснути кнопку "Confirm your email address", в результаті чого система автоматично переведе до impCentral і з'явиться форма реєстрації, де необхідно буде ввести логін, пароль та деяку інформацію про себе (рисунок 2.11).

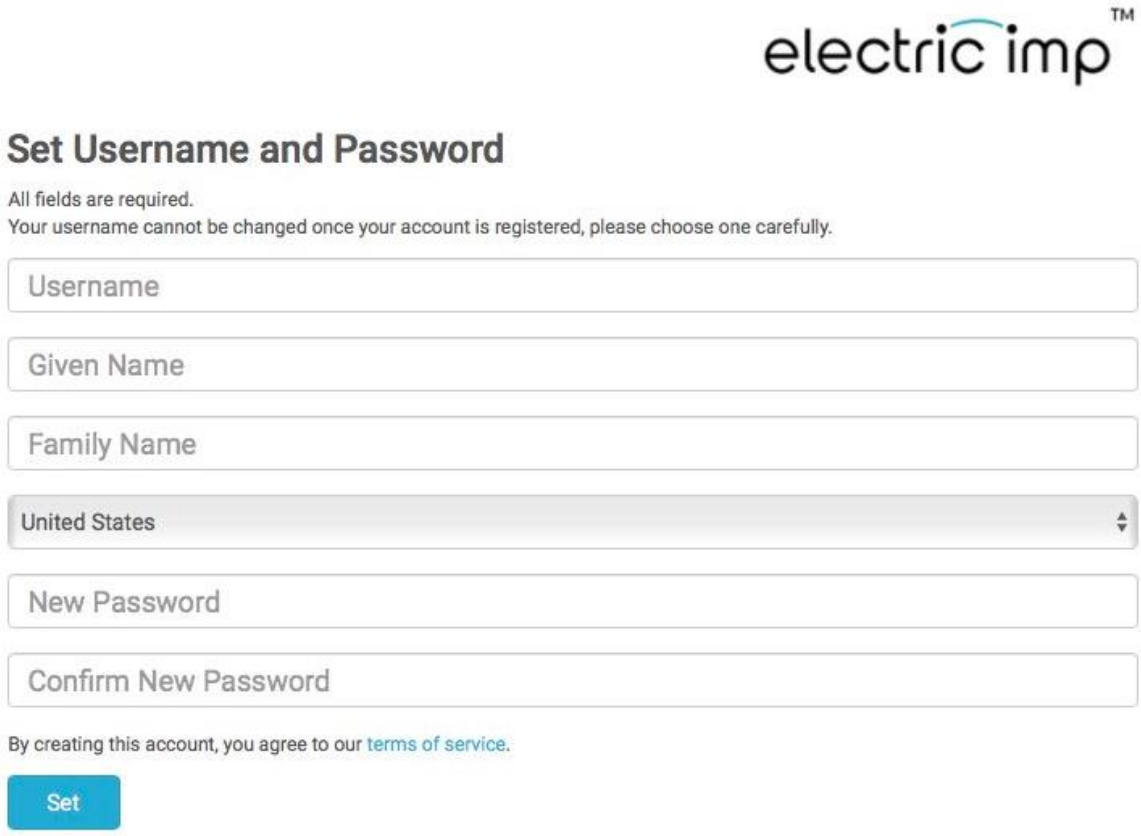

Рисунок 2.11 - Реєстраційна форма impCentral

Для активації імпу необхідно використати запатентовану технологію BlinkUp, для цього треба завантажити Electric Imp додаток з App Store або Google Play. Після того як додаток було встановлено, треба запустити та ввести логін / пароль які були встановлені раніше (рисунок 2.12).

Рисунок 2.12 - Додаток Electric Imp

Далі треба вибрати impCloud, що підтримується імпом - Amazon AWS (рисунок 2.13).

Рисунок 2.13 - Вибір impCloud

Після входу в додаток Electric Imp натисніть «Configure a device», а потім виберіть тип мережі: бездротову або через Ethernet (рисунок 2.14).

Рисунок 2.14 - Вибір типу мережі

Всі імпи підтримують Wi-Fi, але лише imp005 підтримує Ethernet. Використовуйте бездротовий зв'язок, якщо ви плануєте підключити ваш імп через Wi-Fi або Ethernet та вибирайте Ethernet лише в тому випадку, якщо не збираєтеся підключати ваш імп через Wi-Fi. Після вибору бездротового зв'язку, необхідно логін та пароль від мережі та перейти до останнього етапу активації (рисунок 2.15).

Рисунок 2.15 - Останній етап активації за технологією BlinkUp

Статус LED імпу буде зеленого кольору, це означає що імп був успішно активований.

В результаті проведених дій, був створений обліковий запис в середовищі impCentral та активовано імп, для продовження слід перейти до сторінки реєстрації / логіну та ввести створені дані (рисунок 2.16).

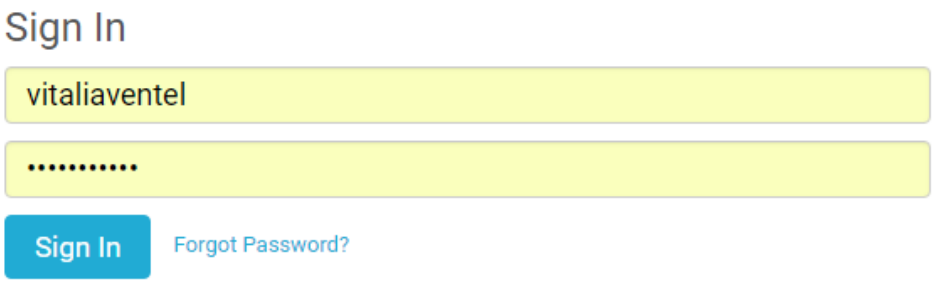

Рисунок 2.16 - Вхід в середовище impCentral

impCentral забезпечує всі інструменти, необхідні для розгортання програмного забезпечення, яке буде керувати вашим продуктом. impCentral запускається у браузері. Він оптимізований для Firefox і Google Chrome, але він буде працювати в інших популярних браузерах на Linux, MacOS і Windows, таких як Safari і Edge.

Далі треба перейти до "My Development Devices". Наразі імп перебуває в статусі "unassigned", що просто означає, що він ще не пов'язаний з будь-якою групою пристроїв impCentral (рисунок 2.17).

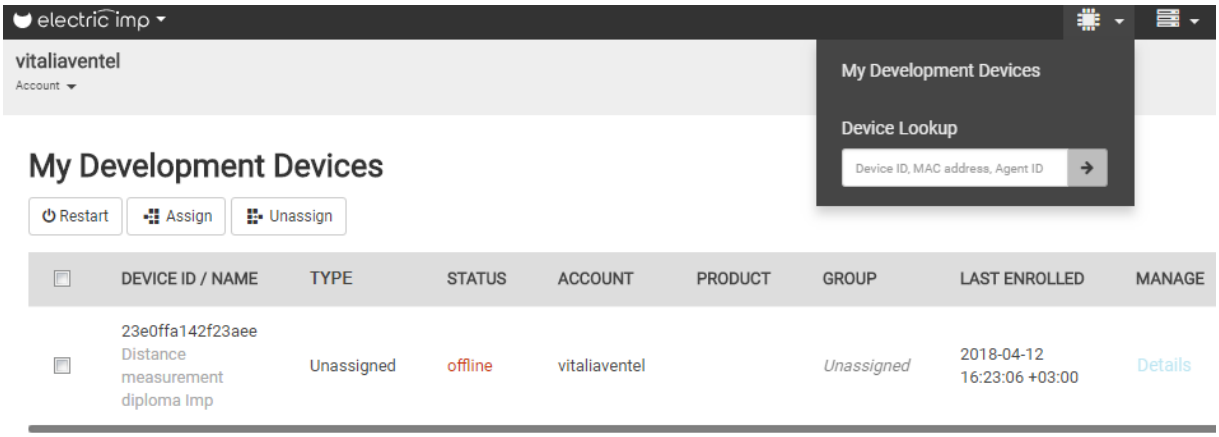

Рисунок 2.17 - My development devices

Перш ніж ми продовжуватимемо створювати програмне забезпечення, яке буде запускати ImpExplorer Developer Kit, необхідно поговорити про фундаментальний аспект термінології Electric Imp: що таке "пристрій" та "імп".

Імп - це компонент, який забезпечує підключення до Інтернету та обчислювальний інтелект до вашого підключеного пристрою. Це може бути карта imp001, як у вашому комплекті для розробника ImpExplorer. Пристрій являє собою комбінацію імпу та інших апаратних компонентів. За допомогою комплекту ImpExplorer, пристрій являє собою комбінацію картки imp001, її головної дошки та будь-якого іншого обладнання, яке ви додаєте, наприклад датчиків, дисплеїв, кнопок та перемикачів.

Оскільки імп - це лише одна частина пристрою, тому саме пристрої представлені в impCentral, а не імпи. Наприклад, є два комплекти розробника impExplorer, обидва вказані в impCentral і налаштовані для запуску різних програм. У будь-який час ви можете обміняти imp001 карти цих пристроїв, але вони будуть працювати правильно. Кожен imp001 автоматично отримує програмне забезпечення, необхідне для забезпечення належного функціонування пристрою. Імп завжди "успадковує" поведінку пристрою, на який він встановлений, і саме тому impCentral зосереджується на пристроях, а не на імпах.

impCentral організовує пристрої та код наступним чином:

- *Product* Це контейнер проекту верхнього рівня. Його зазвичай названо за продукт, який ви створюєте.
    - *Device Group* Це групи пристроїв відповідно до коду, який вони запускають. Усі пристрої, призначені для певної групи пристроїв, запускають один і той же код, який можна переглядати та змінювати редактором коду. Існує чотири типи Device Group:
        - *Development* Місце де ведеться розробка та тестування коду додатку.

- *Factory Test* Місце де ведеться розробка та тестування фабрики, якщо у облікового запису є на це права.
- *Production* Місце де йде розгортання коду додатку, якщо у облікового запису є на це права.
- *Factory* Місце де йде розгортання коду фабрики, якщо у облікового запису є на це права.

Знайдіть свій обліковий запис на навігаційній панелі impCentral, потім натисніть меню "Account" під ним та виберіть свій особистий обліковий запис (рисунок 2.18).

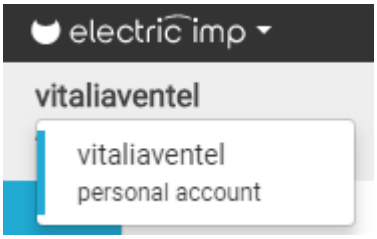

Рисунок 2.18 - Особистий обліковий запис

Це відобразить всі продукти вибраного облікового запису. Натисніть "Create New Product". Тепер на панелі "Create Product" з'явиться ім'я першого продукту, а також назву її першої групи пристроїв розробки. Введіть два імені та натисніть кнопку "Create" (рисунок 2.19).

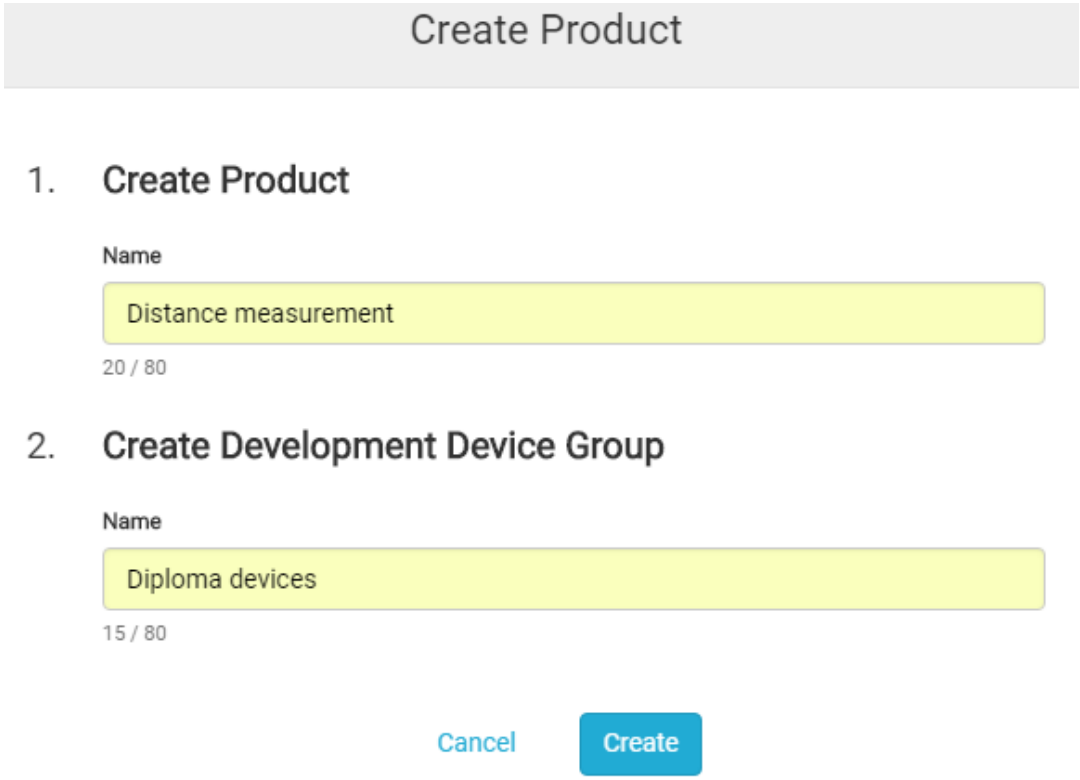

Рисунок 2.19 - Панель "Create Product"

В результаті створення продукта, буде автоматичне перенаправлення до редактора коду, який буде використовуватися для розробки коду додатка. Щоб протестувати цей код, вам потрібне обладнання, таке як імп комплект. Щоб імп комплект запустив код, над яким ви працюєте в групі пристроїв, він повинен бути "призначений" для цієї групи пристроїв. Подивіться на панель логів в нижній частині вікна, і ви побачите кнопку "Assign". Після натискання на кнопку "Assign" з'явиться модальне вікно (рисунок 2.20).

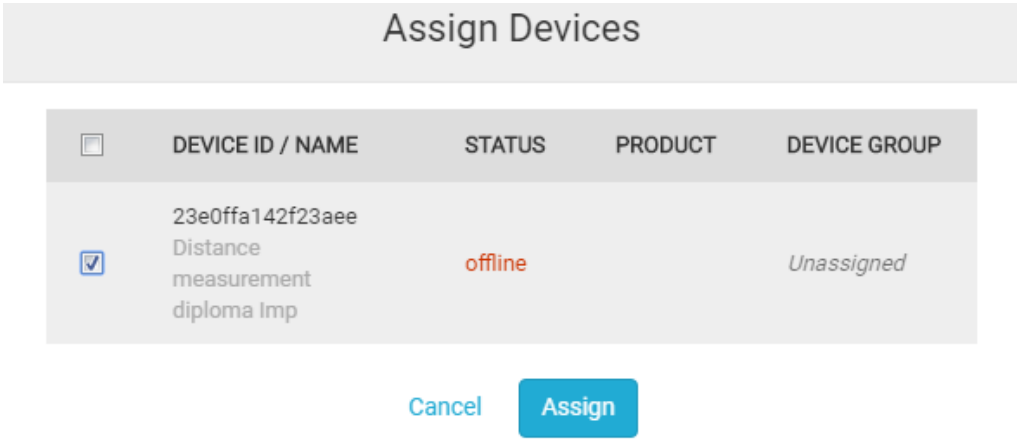

Рисунок 2.20 - Модальне вікно "Assign Devices"

Пристрої розробки назначаються групам пристроїв розробки, і коли призначено, вони завантажують та запускають будь-який код, що розгортається в цій групі пристроїв. Кожна група пристроїв має редактор коду - зміна коду та натискання кнопки "Build and Force Restart" редактора автоматично примушують усі пристрої групи перезавантажуватись та отримувати нове програмне забезпечення (рисунок 2.21).

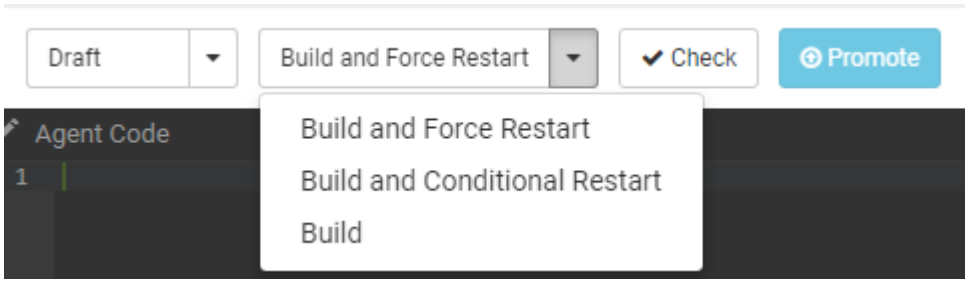

Рисунок 2.21 - Оновлення ПЗ для групи пристроїв

Пристрій програми Electric Imp складається з двох частин. По-перше, це код пристрою, працює на самому комплекті. Другий працює всередині impCloud і називається агентом пристрою. Кожен пристрій має свій унікальний агент, тому, якщо у вас є 10 моделей, у хмарі буде працювати десять агентів, кожен з яких використовує код агента моделі. Кожен пристрій має взаємозв'язок один до одного з його агентом.

Агент зберігає пристрій безпечно: всі зв'язки між пристроєм та зовнішніми службами опосередковуються агентом пристрою. Для забезпечення безпеки відсутній прямий доступ до пристроїв. Це так, навіть якщо зовнішня служба знаходиться в тій же фізичній мережі, що й пристрій. impCloud побудована базуючись на продуктивність, тому будь-яка затримка, яку вносить цей підхід, є незначною.

Агенти постійно працюють, дозволяючи пристроям бути доступними, навіть якщо вони сплять, щоб заощадити енергію. Агент може перевірити стан пристрою та утримувати отримані дані, доки пристрій знову не прокинеться.

2.3 Розробка програмного забезпечення для модуля Electric Imp мовою Squirrel

Кожен імп був розроблений таким чином, щоб забезпечити переваги підключення до Інтернету для різних пристроїв. Так як імпи не спілкуються безпосередньо з хмарою, замість цього це делеговано своїм агентам, тобто мікросервісам на кожному пристрої, що працюють в impCloud. Для керування цим процесом Imp API надає повний набір інструментів.

Мікросервіс знаходить та отримує доступ до інтернет-ресурсів від імені імпу та керує вхідними командами і запитами на інформацію. Цей підхід дозволяє імпу як можна менше часу присвятити на його виконання і сконцентруватись на передачі даних мережею Інтернет, тому агент виконує всю важку роботу.

Так як розробник коду агента запущеного в impCloud та коду пристрою запущеного на імпі є головним в зв'язку між імпом та агентом, окрім того з віддаленими програмами та веб-сайтами.

Так як Electric Imp розроблений спеціально для Інтернету, агент спілкується з хмарою за допомогою протоколу HTTP. Це означає, що він може взаємодіяти з будь-якою програмою, яка може видати стандартні HTTP-запити: локальна або серверна веб-сторінка, мобільний або настільній додаток, написаний для підтримки певної операційної системи. Якщо програмне забезпечення може надсилати запит HTTP на URL-адресу, він може з'єднатися з імпом за допомогою агента.

HTTP є відкритим стандартом, тому він добре визначений і доступний. Він підтримується W3C, який надає повну специфікацію. Специфікація визначає HTTP-повідомлення, відомі дві форми: HTTP-запит і HTTP-відповідь[6].

Найпростішим запитом HTTP є запит на дані, зроблені клієнтом на сервері та отримані за допомогою виклику єдиної URL-адреси. Тим не менш, це можна масштабувати для складних запитів, які передають дані та керують безпекою.

Головним вузлом комунікації є агент. Серверний код агенту реагує на вхідні HTTP-запити, аналізуючи, що вони містять, а потім при необхідності

запускає відповідну логіку в самому імпі. Основний принцип взаємодії app-agent-imp відповідає даному шляху: App -> impCloud -> device (рисунок 2.22).

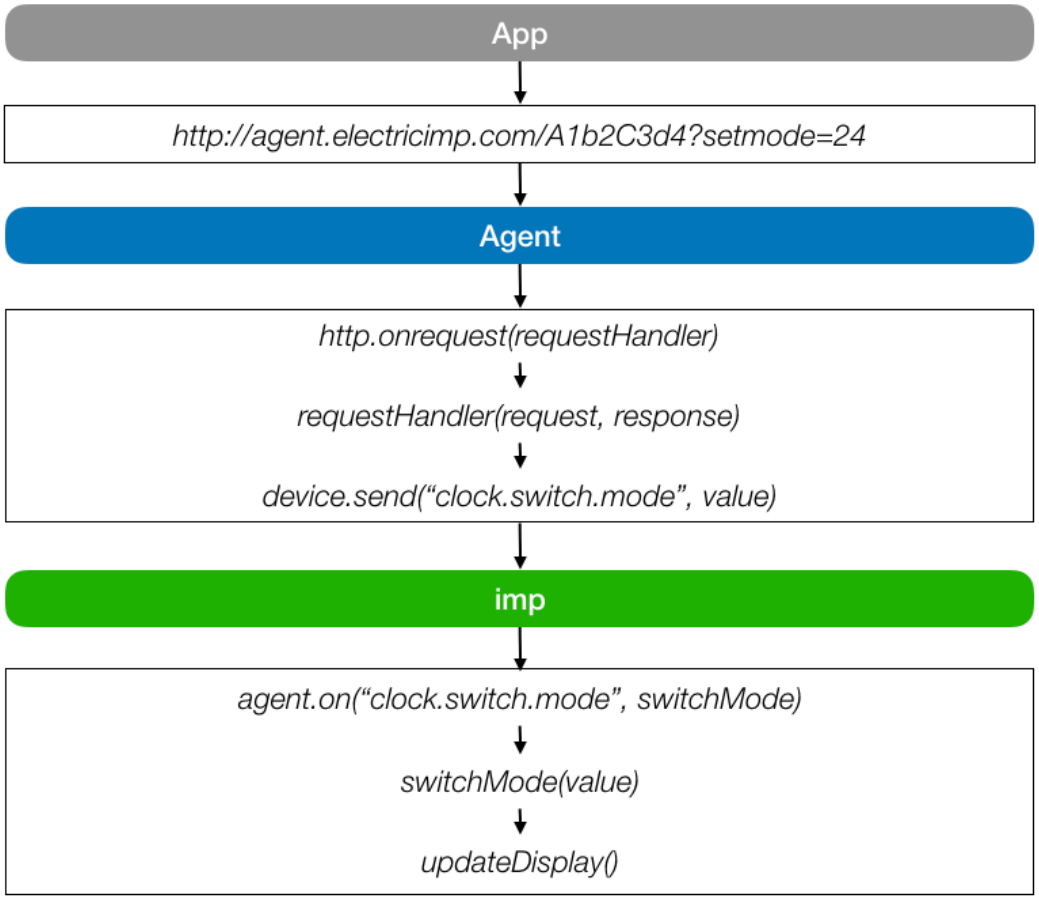

Рисунок 2.22 - Принцип взаємодії app-agent-imp

imp API автоматично ініціалізує http-об'єкт, який надає метод http.onrequest(). Цей метод реєструє ім'я функції, яка буде викликана, коли агент отримує HTTP-запит. Функція зворотного виклику запускається лише тоді, коли відбувається ця подія, і викликається кожного разу, коли відбувається така подія. Метод http.onrequest() - це лише назва зворотного виклику.

Сама функція зворотного виклику розроблюється по best practice: слід обернути код, який оброблює HTTP-запит в структуру try/catch мови Squirrel[7]. Це зроблено для того, щоб забезпечити механізм кодування, який буде повідомляти про зв'язок та інші помилки:

```
function requestHandler(request, response) {
```

```
  try {
    // "200: OK" is standard return message
    response.send(200, "OK");
  } catch (ex) {
    // Send 500 response if error occurred
    response.send(500, ("Agent Error: " + ex));
  }
}
// Register the callback function that will be triggered
by incoming HTTP          requests
http.onrequest(requestHandler);
```

Метод response.send() є членом imp API об'єкта httpresponse. Він приймає два параметри: ціле число, що містить W3C стандартний код HTTP-відповіді та рядок повідомлення. Метод http.onrequest() автоматично генерує об'єкт httpresponse і передає його разом із вихідним вхідним HTTP-запитом до обробника запиту. Цей об'єкт відповіді кешується, тому можна отримати доступ через локальну змінну, до відповіді яка використовується для повернення даних до джерела запиту. Збережена відповідь буде використовуватися як спосіб передачі даних від імпу до додатку.

Сам HTTP-запит - доступ до обробника через локальну змінну, запит - це Squirrel таблиця, яка містить ключові елементи запиту, збережені як набір пар ключ/значення. Ключі походять від стандартної структури запиту HTTP[7]:

- метод (*method*) це тип HTTP запиту, наприклад GET, POST, PUT;
- шлях (*path*) це шлях до вказаного HTTP ресурсу без URL агенту;
- запит (*query*) таблиця Squirrel містить параметри запиту як набір пар ключ/значення;
- тіло (*body*) це тіло HTTP запиту в форматі JSON;
- заголовки (*headers*) таблиця Squirrel містить заголовки запиту як набір пар ключ/значення.

Наприклад, якщо нашому агенту надсилається наступний HTTP-запит у полі URL-адреси браузера:

```
https://agent.electricimp.com/m5Q7ePwvVNcw?rssi=26&tof=101
```

Тоді запит буде зберігати наступну таблицю, зображену на рисунку 2.23.

Key	Value
rssi	26
tof	101

Рисунок 2.23 - Squirrel таблиця пар ключ/значення

Таблична організація даних запиту HTTP дозволяє легко знаходити параметри запиту, читати їх значення та викликати поведінку в Imp. Допоміжні таблиці та їхні значення пар ключ/значення даних доступні за допомогою використання крапки. Використовуючи таблицю request.query, читається параметр rssi, і якщо він є передається початкове значення на пристрій:

```
if ("rssi" in request.query) {
  device.send("rssi.init.value", request.query.rssi);
  response.send(200, "RSSI value initialized");
}
```

Об'єкт device є імпом для агента і дозволяє агенту спілкуватися з ним. Викликається метод device.send(), щоб надіслати повідомлення, на яке імп може відповідати або не відповідати. Першим параметром у device.send() є назва повідомлення, другим - єдине значення даних. Ось ці дані є цілим числом, але

можна надсилати будь-який інший тип даних Squirrel: float, boolean, string, array, table, blob.

Так само, як агент має device об'єкт який є представленням імпу, так імп має agent об'єкт, щоб обробляти події надіслані з сервера. Імп викликає функцію agent.on() для обробки певних повідомлень:

```
agent.on("rssi.init.value", rssiInit);
```

Метод agent.on() приймає назву повідомлення, яке повинно співпадати на обох частинах платформи як на пристрої так і на агенті. Інший його параметр - це назва функції, яка буде викликана при надходженні повідомлення. Значення даних, передане device.send(), автоматично передається на функцію зворотного виклику:

```
rssi <- null;
function rssiInit(initalValue) {
   rssi = initalValue;
}
```

На рисунку 2.24 зображений повний цикл роботи в середовищі Electric Imp.

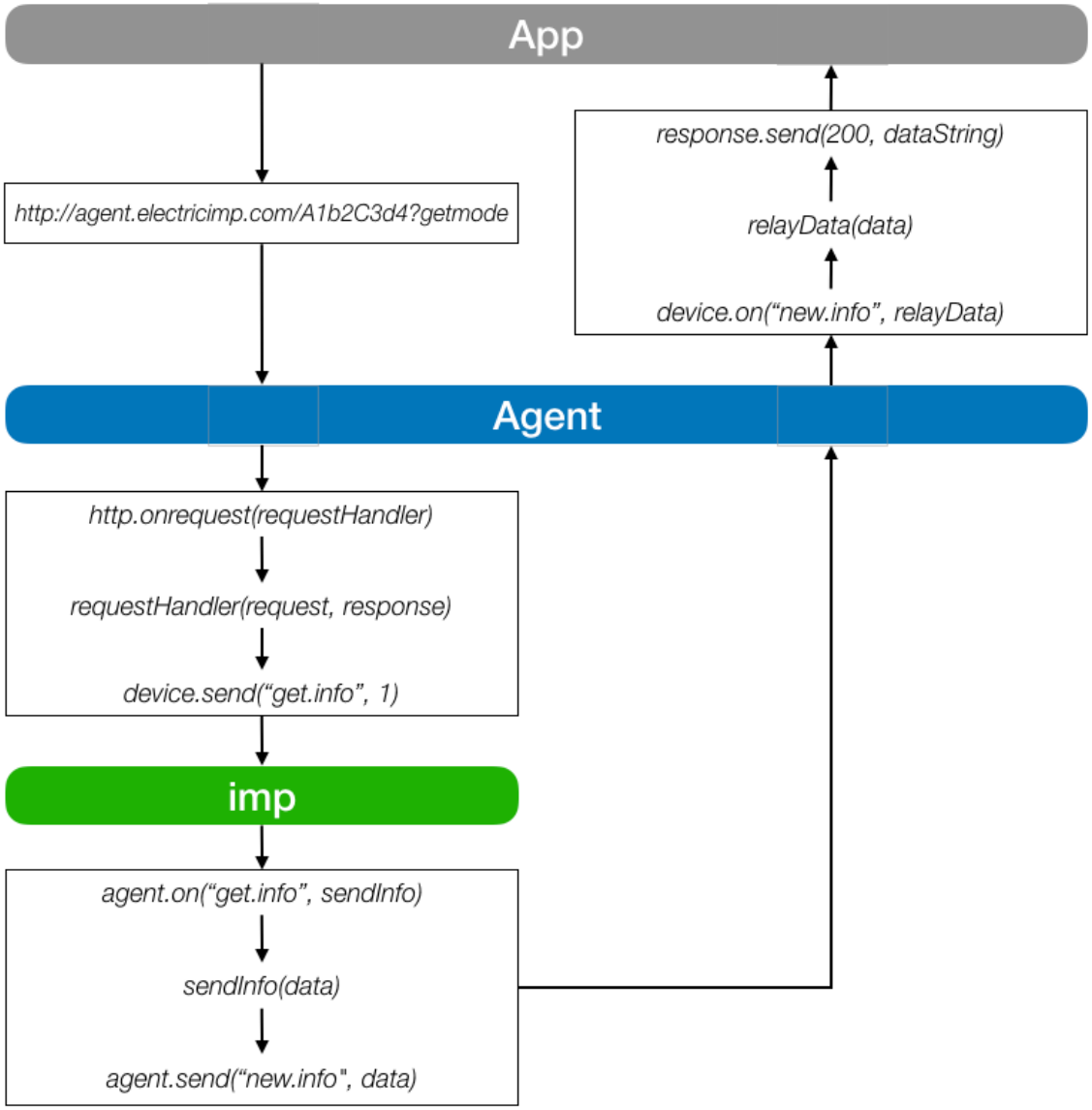

Рисунок 2.24 - Повний цикл Electric Imp

Під час холодного запуску на імпі відбувається наступні дії

Ініціалізація шини I2C.

```
// Configure I2C bus for sensors
local i2c = hardware.i2c89;
i2c.configure(CLOCK_SPEED_400_KHZ);
```

Метод configure() вмикає шину I2C та встановлює тактову частоту. Метод приймає в якості вхідного параметра константу CLOCK_SPEED_400_KHZ, що встановлює значення тактової частоти в 400 kHz.

Ініціалізація датчика температури.

```
tempSensor = HTS221(i2c);
tempSensor.setMode(HTS221_MODE.ONE_SHOT);
```

Створюється об’єкт HTS221, що інкапсулює датчик температури HTS221. Конструктор HTS221 приймає в якості вхідного параметру об’єкт шини I2C. Метод setMode режим зчитування показників. Константа HTS221_MODE.ONE_SHOT визначає, що зчитування буде виконуватися тільки при виклику методу read().

Ініціалізація SPI та світлодіоду.

```
// Configure SPI bus and powergate pin for RGB LED
local spi = hardware.spi257;
spi.configure(MSB_FIRST, 7500);
hardware.pin1.configure(DIGITAL_OUT, 1);
led <- WS2812(spi, 1);
```

Реєстрація на повідомлення «get.temperature» методу takeReading від агенту.

```
agent.on("get.temperature", takeReading);
```

На агенті під час холодного старту відбувається реєстрація на подію відправки HTTP-запиту на зовнішню URL-адресу та реєстрація на повідомлення від імпу «reading.sent».

```
// Register a function to receive sensor data from the
device
device.on("reading.sent", motion);
```

```
// Register the HTTP handler to begin watching for HTTP
requests from your browser
http.onrequest(requestHandler);
```

Під час запиту на зовнішню адресу викликається метод requestHandler.

```
function requestHandler(request, response) {
  try {
    Gresponse = response;
    device.send("get.temperature", null);

  } catch (ex) {
    response.send(500, "Internal Server Error: " + ex);
  }
}
```

Метод requestHandler зберігає параметр response у глобальній змінній та відправляє на імп повідомлення «get.temperature». Отримавши повідомлення на імпі викликається метод takeReading, в якому зчитується показник датчику температури та відправляється повідомлення «reading.sent» та результати виміру на агент.

```
function takeReading(ttr){
  local reading = tempSensor.read();

  // Send 'conditions' to the agent
  agent.send("reading.sent", reading.temperature);

  led.set(128, [0,128,128]).draw();
   imp.wakeup(0.1, shutDownLed);
}
```

Після отримання повідомлення на агенті викликається метод motion(), в якому виконується перетворення числового значення в рядок та відправка відповіді на HTTP-запит за допомогою методу send().

```
function motion(data) {
  local message = format("%.1f", data);
  ::Gresponse.send(200, message);
  server.log(message);
}
```

Для перевірки роботи імпу скористаємося утилітою cURL (рис. 2.25).

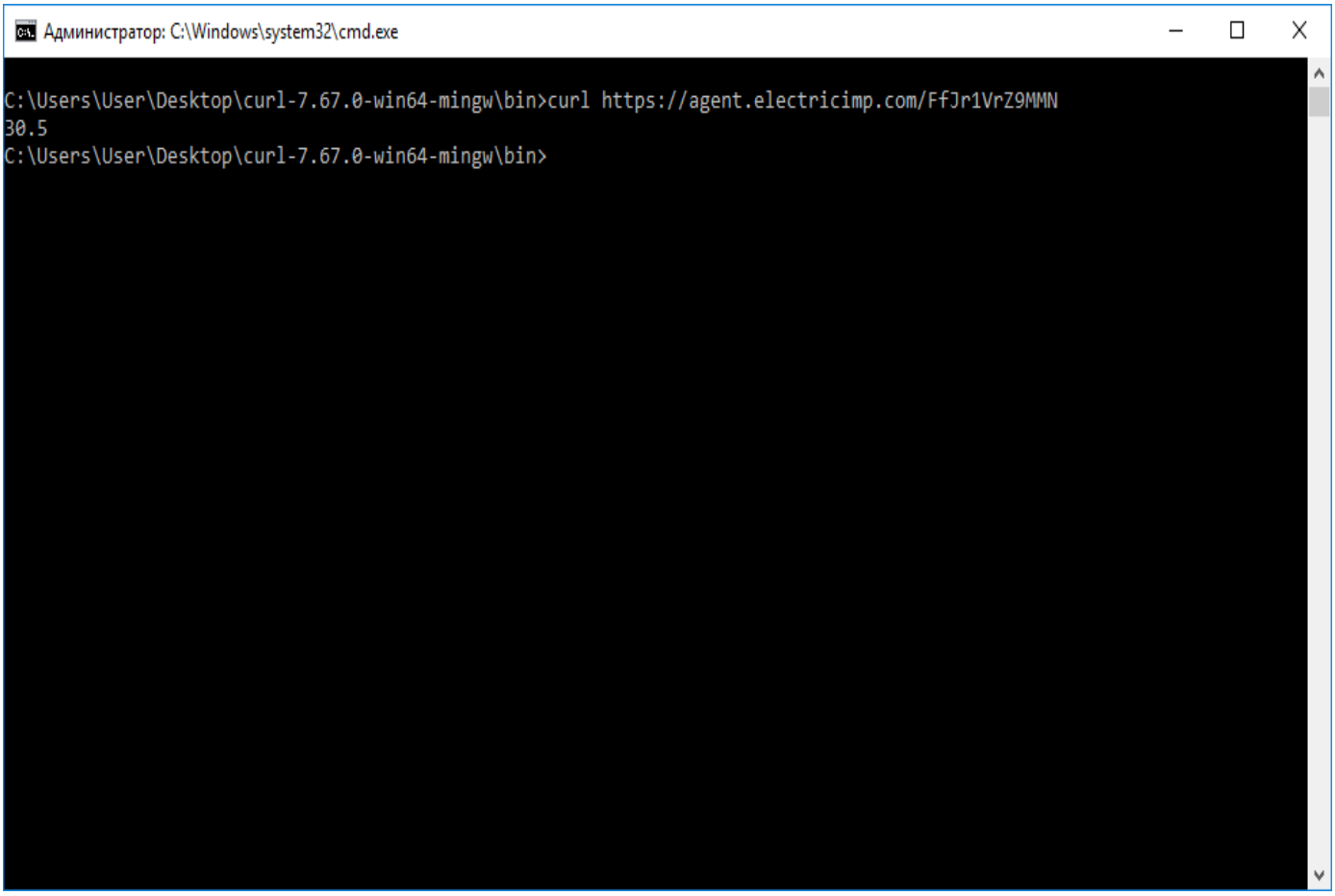

Рисунок 2.25 – Результат запиту на URL агента

Висновки

В розділі надано докладний опис особливостей платформи Electric Imp, описані можливості та функції платформи. Було проведено детальний огляд комплекту impExplorer та Imp001. Досліджено два режими роботи операційної системи impOS. Розглянуто головні етапи в налаштуванні проекту в середовищі impCentral: реєстрація в impCentral, активація імпу за допомогою BlinkUp технології та створення продукту та пристрою в даному середовищі. Описано розробку коду для агенту та пристрою в середовищі розробки impCentral за допомогою мови Squirrel. Описана основна парадигма використовувана при розробці ПЗ для Electric Imp. Задачею даного модуля є відправка результату виміру температури в якості відповіді на HTTP-запит.

3 МОДЕЛЮВАННЯ ПОДІЄВО-КЕРОВАНИХ ВИМІРЮВАЛЬНИХ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ З ВИКОРИСТАННЯМ ПАКЕТУ GPSS WORLD

Моделювання є універсальним методом отримання та використання знань про навколишній світ. Моделювання завжди використовується людиною в цілеспрямованій діяльності, особливо в дослідницькій. В сучасних умовах посилюється роль і значення математичного моделювання, яке з розвитком засобів обчислювальної техніки часто стали називати комп'ютерним[8].

Математичні (комп'ютерні) моделі, в силу своєї логічності і суворого формального характеру, дозволяють виявити основні фактори, що визначають властивості досліджуваних систем і досліджувати їх реакції на зовнішні впливи і зміни параметрів. Часто математичні моделі простіше і зручніше використовувати, ніж натуральні (фізичні). Вони дозволяють проводити обчислювальні експерименти, реальна постановка яких утруднена або неможлива. В розділі розглянуто розробка моделі подієво-керованої вимірювальної системи в пакеті імітаційного моделювання GPSS World.

3.1 Огляд методів моделювання

Моделювання – це заміщення вихідного об’єкта іншим об’єктом, що називається моделлю, та проведення експериментів з моделлю з метою отримання нової інформації про вихідний об’єкт.

Модель – це фізичний або абстрактний об’єкт, що адекватно відображає досліджувану систему.

За ступенем абстрагування моделі від оригіналу моделі поділяють на дві групи: матеріальні(фізичні) та абстрактні(математичні).

Фізичною моделлю зазвичай називають систему, яка еквівалентна або подібна оригіналу або у якій процес функціонування такої ж, як у оригіналу, і має ту ж або іншу фізичну природу.

Математична модель – формалізований опис системи за допомогою абстрактної мови, зокрема математичних співвідношень, що відображають функціонування системи. В зв’язку з обчислювальними можливостями сучасних ЕОМ широкого розповсюдження набуло математичне моделювання.

В залежності від цілей моделювання може проводитися на двох рівнях: на якісному та на кількісному. Відповідно застосовуються моделі: наглядні та конструктивні. Математичне моделювання проводиться на кількісному рівні. При досліджені технічних систем найбільш широко використовуваними методами є: аналітичні, чисельні, статистичні або імітаційні та комбіновані.

Аналітичні методи полягають у побудові математичної моделі у вигляді математичних символів та відношень. Перевагою аналітичних методів полягає в можливості отримання рішення в явній аналітичній формі. Перевагою аналітичних методів є отримання результату в явній аналітичній формі, що дозволяє проводити детальний аналіз процесів, що відбуваються в досліджуваній системі, в широкому діапазоні параметрів системи. Недоліком аналітичних методів – це використання цілого ряду припущень та неможливість, у деяких випадках, отримати результат через нерозв'язності рівнянь в аналітичній формі.

Чисельні методи базуються на побудові послідовності дій над числами. Застосування чисельних методів зводиться до заміни математичних операцій і відносин відповідними операціями над числами, наприклад, до заміни інтегралів сумами, нескінченних сум - кінцевими і т.п. Результатом застосування чисельних методів є таблиці і графіки залежностей, які розкривають властивості об'єкта. Чисельні методи є продовженням аналітичних методів в тих випадках, коли результат не може бути отриманий в явному вигляді. Чисельні методи порівняно з аналітичними методами дозволяють вирішувати значно більш широке коло завдань.

Імітаційні методи полягають у відтворені на ЕОМ процесу функціонування системи, дотримуючись часової та логічної послідовності протікання процесів. Імітаційне моделювання дозволяє отримати дані про

систему у довільні моменти часу[9]. Для імітації процесу формують алгоритм, що дозволяє проводити обчислювальні експерименти.

Серед властивостей об’єкту варто розрізняти вплив на об’єкт та його реакцію на вплив. Кількісне вираження цих величин здійснюється за допомогою параметрів. Будь-який процес або об’єкт може бути зображеним наступним чином (рисунок 3.1)

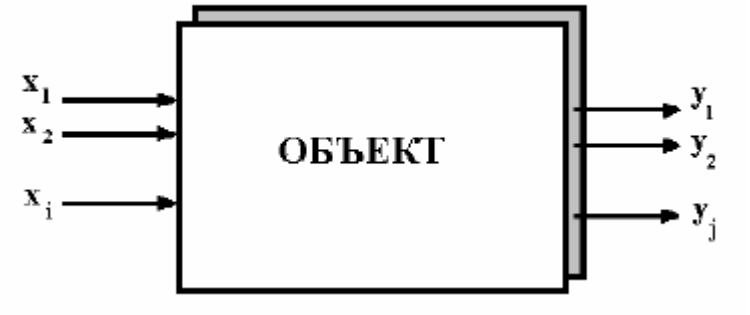

Рисунок 3.1 – Схематичне зображення об’єкту моделювання де x_i – вхідні параметри, а y_i – вихідні

Вхідні параметри об’єкту поділяють на:

- Зовнішні параметри – параметри значення яких можуть бути змінені, але можливості впливати на них немає;
- Керуючі параметри - параметри на які можна здійснювати прямий вплив відповідно до вимог, що дозволяє керувати системою;
- Збуджуючі параметри – параметри що змінюються довільним чином та на доступні для змін.

Перед математичними моделями ставляться вимоги універсальності, точності, адекватності та економічності.

Ступінь універсальності математичними моделі характеризує повноту відображення моделлю властивостей реального об’єкту. Зазвичай в моделі описуються не всі властивості, наприклад, в деяких дослідженнях не вимагається щоб модель описувала колір або форму об’єкту. Універсальність моделі відображає її застосовність до широкого класу об’єктів. Ступінь універсальності не має кількісної характеристики.

Точність математичної моделі характеризує ступінь співпадіння значень параметрів реального об’єкту та моделі. Точність характеризується відносною похибкою.

Адекватність математичної моделі – здатність відображати властивості об’єкту з похибкою в більше заданої.

Економічність математичної моделі характеризується затратами обчислювальних ресурсів (процесорного часу та пам’яті).

3.2 Загальна методика створення математичних моделей

Створення математичної моделі складається з чотирьох етапів:

1. Постановка задачі. На цьому етапі визначають які властивості об’єкту будуть відображені в моделі та властивості, що вважаються не суттєвими. Вибір властивостей, що відображаються в моделі базується на аналізі можливих областей застосування моделі та визначає її ступінь універсальності;
2. Синтез математичної структури моделі. Цей етап полягає в отримані загального виду математичних співвідношень без конкретизації числових значень які фігурують в них;
3. Визначення числових значень параметрів моделі.
4. Аналіз моделі. На цьому етапі проводять оцінку адекватності та точності математичної моделі.

Якщо в ході аналізу моделі з’ясовують, що вона не задовільно відображає об’єкт, то намагаються скоригувати числові параметри. В випадку коли коригування не дало бажаного ефекту, варто скоригувати структуру моделі та повторити наступні два етапи. Якщо після цього не задовільно описує об’єкт, то помилка була допущена на стадії постановки задачі і необхідно заново створювати модель.

Дана методика передбачає, що деякі етапи можуть виконуватись неодноразово в процесі наближення до бажаного результату. Етап постановки задачі є основоположним та найважливішим етапом, що базується на знаннях про об’єкт дослідження та не піддається формалізації. Інші етапи створення математичної моделі та робота з нею допускають використання методів обчислювальної математики та можуть бути формалізовані. Тому вони повинні виконуватись з використанням сучасних засобів обчислювальної техніки. Для цього необхідно створити відповідні алгоритми та розробити програми для ЕОМ.

3.3 Імітаційне моделювання

Імітаційне моделювання полягає у логічно-аналітичній імітації функціонування системи. При цьому для детермінованих систем визначаються зміни їх стану під впливом зовнішніх дій[10]. А для стохастичних систем, окрім інформації про зміни стану системи, отримують вибірки значень вихідних параметрів за якими визначаються їх основні імовірнісні характеристики.

На відміну від аналітичних моделей, які передбачають наявність математичного опису процесів, що протікають в оригіналі і які зазвичай будуються при жорстких обмеженнях на параметри, імітаційні моделі є більш універсальними і можуть бути побудовані за відсутності математичного опису оригіналу. Ідея імітаційного моделювання дуже проста і полягає в тому, що будується якийсь алгоритм поведінки під- систем і окремих елементів систем в часі. Цей алгоритм може бути реалізований у вигляді програми для ЕОМ. Багаторазово «проганяючи» імітаційну модель (ІМ) в умовах випадкових потоків подій на вході і в самій системі, можна накопичити статистичну інформацію про зміну змінних стану ІМ. Статистична обробка цієї інформації дозволяє отримати статистичні оцінки показників ефективності.

Модель системи завжди являє собою сукупність моделей елементів і їх функціональні взаємозв'язки. Модель елемента системи - це зазвичай набори правил (алгоритмів) поведінки елемента по відношенню до вхідних впливів і

змін станів елемента. Якщо елемент відображає функціональний прилад на тому чи іншому рівні деталізації, то в найпростішому випадку він може знаходитись у робочому стані або у неробочому. У робочому стані елемент може бути зайнятим або вільним.

Основними цілями імітаційного моделювання Є:

- Відтворення з необхідною достовірністю поведінки окремих елементів системи в процесі реалізації нею функції системи;
- Накопичення статистичних даних про поведінку елементів;
- Наступна статистична обробка цих даних для отримання статистичних оцінок, кількісних характеристик, законів розподілу оцінюваних показників ефективності.

Імітаційна модель - універсальний засіб дослідження складних систем, що представляє собою логіко-алгоритмічний опис поведінки окремих елементів системи та правил їх взаємодії, що відображають послідовність подій, що виникають в моделюється системі[11].

Поняття «статистичне і імітаційне моделювання» часто розглядають як синоніми. Однак слід мати на увазі, що статистичне моделювання не обов'язково є імітаційним. Наприклад, обчислення певного інтеграла методом Монте-Карло шляхом визначення підінтегральної площі на основі безлічі статистичних випробувань, відноситься до статистичного моделювання, але не може називатися імітаційним.

Найбільш широке застосування імітаційне моделювання отримало при дослідженні складних систем з дискретним характером функціонування, в тому числі моделей масового обслуговування. Для опису процесів функціонування таких систем зазвичай використовуються часові діаграми.

Часова діаграма - графічне представлення послідовності подій, що відбуваються в системі. Для побудови часових діаграм необхідно досить чітко уявляти взаємозв'язок подій всередині системи. Ступінь деталізації при складанні діаграм залежить від властивостей модельованої системи і від цілей моделювання.

Оскільки функціонування будь-якої системи досить повно відображається у вигляді часової діаграми, імітаційне моделювання можна розглядати як процес реалізації діаграми функціонування досліджуваної системи на основі відомостей про характер функціонування окремих елементів і їх взаємозв'язку.

Імітаційне моделювання зазвичай проводиться на ЕОМ відповідно до програми, що реалізує заданий конкретний логіко алгоритмічний опис. При цьому кілька годин, тижнів або років роботи досліджуваної системи можуть бути промоделювати на ЕОМ за кілька хвилин. У більшості випадків модель є не точним аналогом системи, а скоріше її символічним відображенням. Однак така модель дозволяє проводити вимірювання, які неможливо провести будь-яким іншим способом.

Імітаційне моделювання забезпечує можливість випробування, оцінки та проведення експериментів з досліджуваної системою без будь-яких безпосередніх впливів на неї.

Першим кроком при аналізі будь-якої конкретної системи є виділення елементів, і формулювання логічних правил, які керують взаємодією цих елементів. Отримане в результаті цього опис називається моделлю системи. Модель зазвичай включає в себе ті аспекти системи, які становлять інтерес або потребують дослідження.

Оскільки метою побудови будь-якої моделі є дослідження характеристик модельованої системи, в імітаційну модель повинні бути включені засоби збору та обробки статистичної інформації з усіх характеристик, що базуються на методах математичної статистики.

Процес функціонування такої системи може бути представлений у вигляді часових діаграм, на основі яких можуть бути виміряні і розраховані характеристики обслуговування заявок. Оскільки процеси надходження і обслуговування заявок в системі носять випадковий характер, то для побудови діаграм необхідно мати генератори випадкових чисел.

У нашому розпорядженні є генератори випадкових чисел, що формують значення відповідних випадкових величин із заданими законами розподілів

A(τ) і B(τ). Тоді можна побудувати часові діаграми, що відображають процес функціонування даної системи (рис. 3.2).

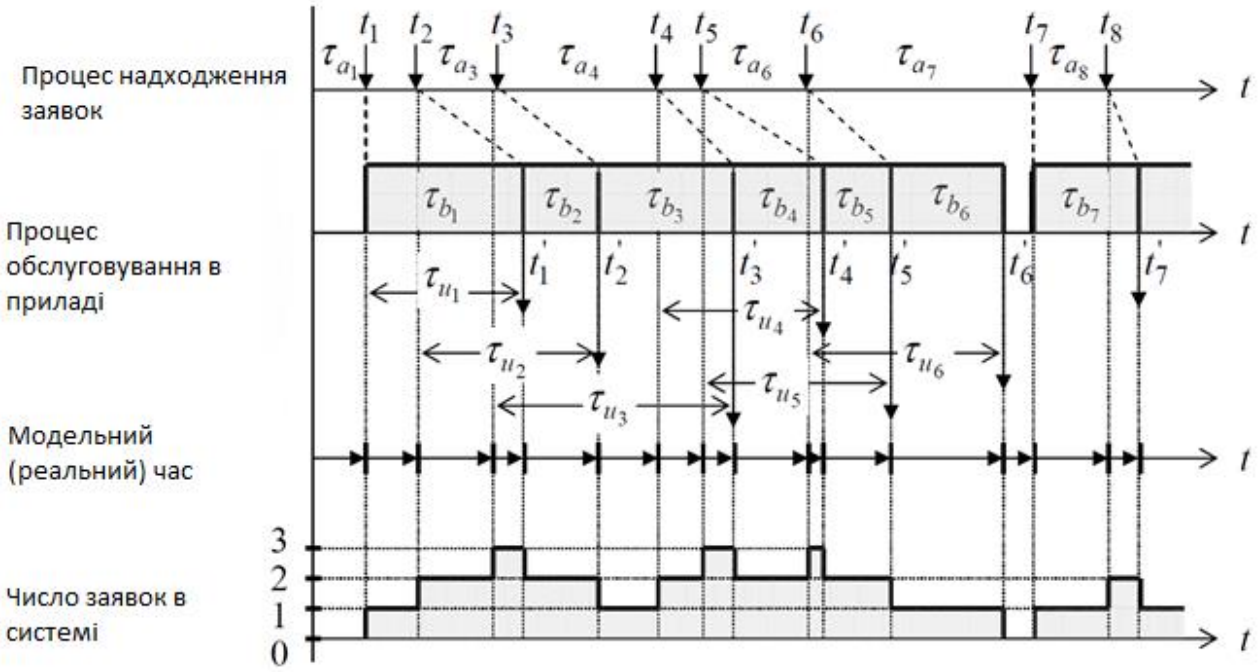

Рисунок 3.2 - Діаграми функціонування одноканальної СМО

На рисунку 3.2 зображені діаграми, що відображають:

- Процес надходження заявок – в вигляді моментів t_i надходження заявок у систему, що формуються за правилом: $t_i = t_{i-1} + \tau_{a_i}\ (t_0 = 0)$, де $\tau_{a_i}\ (i = 1,2,\dots.)$ – інтервали між вступниками в систему заявками, значення яких виробляються за допомогою генератора випадкових величин A(τ);
- Процес обслуговування в приладі, представлений у вигляді тривалостей обслуговування τ_{b_i}, яких виробляються за допомогою генератора випадкових величин B(τ), і моментів завершення обслуговування t_i' заявок в приладі, що визначаються за наступним правилом:
  - $t_i' = t_i + \tau_{b_i}$, якщо на момент надходження i-ї заявки обслуговуючий прилад був вільний;

- $t_i' = t_{i-1}' + \tau_{b_i}$, якщо на момент надходження і-ї заявки обслуговуючий прилад був зайнятий обслуговуванням попередньої заявки.

- «Модельний або реальний час», що показує дискретне (стрибкоподібне) зміну часу в реальній системі, кожен момент якого відповідає одному з наступних подій: надходження заявки в систему або завершення обслуговування заявки в приладі; відзначимо, що в ці моменти часу відбувається зміна стану системи, що описується числом заявок, що знаходяться в системі;
- Число заявок в системі, що описує стан дискретної системи і змінюється за правилом: збільшення на 1 в момент надходження заявки в систему і зменшення на 1 в момент завершення обслуговування.

При дотриманні обраного часового масштабу представлені діаграми дозволяють шляхом вимірювання визначити значення ймовірносно-часових характеристик функціонування модельованої системи, зокрема, як показано на другий діаграмі, час знаходження (перебування) кожної заявки в системі.

час перебування заявок в системі - величина випадкова. У найпростішому випадку, при застосуванні методів математичної статистики, можна розрахувати два перших моменту розподілу часу перебування:

- Математичне очікування: $\mu = \frac{1}{N}\sum_{i=1}^{N} \tau_{u_i}$
- Другий початковий момент: $\mu^{(2)} = \frac{1}{N-1}\sum_{i=1}^{N} \tau_{u_i}^2$

де N - кількість значень часу перебування заявок, отриманих на діаграмі, тобто кількість заявок, відображених на діаграмі як пройшли через систему і покинули її.

На основі отриманих за допомогою часових діаграм значень часу перебування заявок в системі можна побудувати гістограму функції або щільності розподілу часу перебування.

Точність отриманих числових моментів розподілу і якість гістограм істотно залежить від кількості значень N часу перебування заявок, на основі

яких вони розраховуються: чим більше N, тим точніше результати розрахунку. Значення N може становити від декількох тисяч до десятків мільйонів. Конкретне значення N залежить від багатьох факторів, які впливають на швидкість збіжності результатів до істинного значення, основними серед яких при моделюванні систем і мереж масового обслуговування є закони розподілів інтервалів між вступниками заявками і тривалості обслуговування, завантаження системи, складність моделі, кількість класів заявок і т. д.

Побудова вручну таких часових діаграм з тисячами і більш проходять через систему заявками, нереально. У той же час, використання ЕОМ для реалізації часових діаграм дозволяє істотно прискорити процеси моделювання і отримання кінцевого результату. Тому, як сказано вище, імітаційне моделювання можна розглядати як процес реалізації діаграми функціонування досліджуваної системи[12].

У найпростішому випадку тимчасова діаграма може бути реалізована наступним чином: спочатку формуються моменти надходження всіх заявок в систему, а потім для кожної заявки визначаються тривалості обслуговування в приладі і формуються моменти завершення обслуговування (виходу заявок з системи). Очевидно, що такий підхід є неприйнятним, оскільки навіть для дуже простої системи доведеться зберігати в пам'яті ЕОМ одночасно мільйони значень моментів надходження і завершення обслуговування заявок, а також інших змінних, причому зі збільшенням кількості класів заявок і кількості обслуговуючих приладів це число збільшиться багаторазово.

Другий підхід, який може бути запропонований для реалізації часової діаграми, - покрокове побудова діаграми. Для цього слід сформувати змінну для модельного часу і вибрати крок Δt його зміни. У кожен такий момент часу необхідно перевіряти, яка подія (надходження в систему або завершення обслуговування заявки) відбулося в системі за попередній інтервал Δt.

Цей підхід значно скорочує потребу в пам'яті, оскільки в цьому випадку в кожен момент часу необхідно зберігати в пам'яті ЕОМ значення параметрів

(моментів надходження і завершення обслуговування) тільки тих заявок, які знаходяться в системі на даний момент часу.

Недоліки такого підходу очевидні. По-перше, проблематичним є вибір довжини інтервалу Δt. З одного боку, інтервал Δt повинен бути якомога менше для зменшення методичної похибки моделювання, з іншого боку, інтервал Δt повинен бути якомога більше для зменшення часу моделювання.

Найбільш ефективним підходом визнаний підхід зі змінним кроком просування модельного часу, який реалізується відповідно до принципу «до найближчого події». Принцип «просування модельного часу до найближчого події» полягає в наступному. По всіх процесах, паралельно протікає в досліджуваній системі, в кожен момент часу формуються моменти настання «найближчого події в майбутньому». Потім модельне час просувається до моменту настання найближчого з усіх можливих подій. Залежно від того, яка подія виявилося найближчим, виконуються ті чи інші дії. Якщо найближчим подією є надходження заявки в систему, то виконуються дії, пов'язані з заняттям приладу за умови, що він вільний, і занесення заявки в чергу, якщо прилад зайнятий. Якщо ж найближчим подією є завершення обслуговування заявки в приладі, то виконуються дії, пов'язані з визволення приладу і вибором на обслуговування нової заявки з черги, якщо остання не порожня. Потім формується новий момент настання цього ж події. На третій діаграмі «Модельне (реальне) час» (Рисунок 2) просування часу відповідно до цього принципу показано у вигляді стрілок.

Для того щоб забезпечити правильну тимчасову послідовність подій в імітаційної моделі, використовуються системні годинник, що зберігають значення поточного модельного часу. Зміна значення модельного часу здійснюється відповідно до принципу «перерахунку часу до найближчого події». Відзначимо, що одиниці часу в моделі не обов'язково повинні бути конкретними одиницями часу, такими як секунда або годину. Основною одиницею часу в моделі можна вибрати будь-яку одиницю, яка дозволить отримати необхідну точність моделювання. Важливо пам'ятати, одиниці часу

вибираються виходячи з вимог користувача до точності моделювання. Яка б одиниця не була обрана, наприклад мілісекунда або одна десята години, вона повинна незмінно використовуватися у всій моделі.

Крім розглянутої служби часу в імітаційної моделі необхідно реалізувати процедури, пов'язані з формуванням потоків заявок і імітацією обслуговування, з організацією черг заявок, з організацією збору та статистичної обробки результатів моделювання.

Таким чином, імітаційне моделювання дискретних систем з стохастичним характером функціонування, таких як системи і мережі масового обслуговування, передбачає використання ряду типових процедур, що забезпечують реалізацію відповідних імітаційних моделей.

3.4 Аналіз пакету GPSS World

GPSS (General Purpose Simulation System) - система імітаційного моделювання (CIM) загального призначення, призначена для розробки моделей складних систем з дискретним і безперервним характером функціонування і проведення експериментів з метою вивчення властивостей і закономірностей процесів, що протікають в них, а також вибору найкращого проектного рішення серед декількох можливих варіантів[13].

Серед безлічі реалізацій GPSS однією з найбільш доступних і популярних є GPSS World для роботи на персональних комп'ютерах під управлінням ОС Windows. GPSS World має зручний багатовіконний користувацьким інтерфейсом, вбудованими засобами візуалізації та інтерактивного управління процесом моделювання, велику бібліотеку вбудованих процедур, що включає, в тому числі, генератори випадкових величин для більш ніж двох десятків імовірнісних розподілів. Все це робить процес моделювання ефективним і наочним.

У GPSS World включені спеціальні засоби для моделювання великого класу дискретних систем з стохастичним характером функціонування, зокрема,

систем і мереж масового обслуговування, що дозволяє зробити моделі ясними і лаконічними.

3.4.1 Склад системи імітаційного моделювання GPSS World

Система імітаційного моделювання GPSS World містить:

- Мову GPSS - високорівнева мова імітаційного моделювання;
- мова PLUS (Programming Language Under Simulation) - вбудований в GPSS мову програмування низького рівня;
- Компілятор - програма для трансляції (перекладу) з мови високого рівня на мову комп'ютера.

Об'єктами CIM GPSS World є:

- «Модель» або «GPSS-модель» - програму, що написана на мові GPSS і що являє собою послідовність операторів, що описують логіку роботи системи, що моделюється, кожен з яких реалізує деяку конкретну функцію.
- «Процес моделювання» - безпосередньо виконуваний об'єкт, створюваний в результаті трансляції об'єкта «GPSS-модель»; реалізація «процесу моделювання» полягає в переміщенні в моделі деяких рухомих об'єктів, що називаються транзактами.
- «Звіт» - створюється автоматично після закінчення процесу моделювання і містить результати моделювання.
- «Текстовий об'єкт» - текстові файли, які використовуються для спрощення розробки великих моделей і формування бібліотеки вихідних текстів.

Три перших об'єкта є основними і завжди використовуються при імітаційному моделюванні.

3.4.2 Елементи мови GPSS

Елементами мови GPSS World є:

- Алфавітно-цифрові символи: латинські великі та малі літери від «A» до «Z» і цифри від 0 до 9;
- Ім'я - сукупність алфавітно-цифрових символів (від 1 до 200), що починається завжди з алфавітного символу, причому допускається використання букв тільки латинського алфавіту; для того щоб ім'я не співпало з зарезервованими ключовими словами (назвами операторів, системними числовими атрибутами і т.д.), рекомендується використання символу «_» (підкреслення);
- Мітка - ім'я, розташоване в поле мітки оператора для завдання імені об'єкта GPSS-моделі (пам'яті, таблиці, змінної, ...) Або для позначення місця розташування блоку;
- Змінна користувача - ім'я, що використовується в процесі моделювання для зберігання числових і строкових величин;
- Числа – можуть бути цілочисельними та дійсними;
- Системні числові атрибути (СЧА) - змінні, що описують стан процесу моделювання, автоматично підтримувані в GPSS і доступні протягом всього процесу моделювання;
- Арифметичні оператори - задають арифметичні операції
- Оператори відносини - задають логічні умови
- Логічні оператори - задають логічні операції
- Вирази - частина мови PLUS: являють собою сукупність змінних, чисел і СЧА, пов'язаних арифметичними операторами, логічними операторами і операторами відносини; можуть використовуватися в операндах операторів GPSS і в PLUS-процедурах; завжди полягають в круглі дужки;
- Процедури - програми на мові PLUS (PLUS-процедури), вбудовані в GPSS World (стандартна процедура) або створені користувачем (для користувача процедура); звернення до процедури здійснюється шляхом

завдання в якості операнда GPSS-операторів імені процедури з її параметрами;

3.4.3 Склад і структура GPSS-моделі

GPSS-модель являє собою програму, написану на мові GPSS у вигляді послідовності операторів, що описують логіку роботи системи, що моделюється.

Оператори GPSS-моделі діляться на дві групи: GPSS-оператори та PLUS-оператори. GPSS-оператори діляться на команди і блоки.

Команди призначені для опису об’єктів(змінні, функції, матриці) та для управління процесом моделювання(запуск, зупинка і продовження процесу моделювання, скидання статистики, завершення моделювання).

Оператори блоків або просто блоки представляють собою виконувані оператори і реалізують в процесі моделювання деякі дії, запропоновані цими операторами[14]. Назва «блок» походить від так званих блок-діаграм, які використовуються для графічного представлення GPSS-моделей. Блок-діаграма являє собою набір стандартних блоків, пов'язаних між собою в послідовності, яка визначається логікою роботи модельованої системи. Кожному такому блоку в мові GPSS відповідає певний оператор, який реалізує деяку конкретну функцію. При цьому потрібно було, що реалізація оператора пов'язана з виконанням досить великий сукупності дій (блоку). У GPSS World зображення цих блоків і переміщення транзактів в процесі моделювання можна побачити у вікні «BLOCK ENTITIES».

Система імітаційного моделювання GPSS, призначена для моделювання складних систем зі стохастичним характером функціонування, володіє ефективними вбудованими засобами для опису паралельних процесів, що протікають в досліджуваній системі. Ці засоби дозволяють виконувати опис кожного з паралельних процесів незалежно один від одного у вигляді окремих програмних модулів, що істотно спрощує процес програмування і створення моделі. У той же час, якість моделі в значній мірі залежить від того, наскільки

достовірно і коректно відображені в моделі зв'язку між модулями, що відображають логіку функціонування модельованої системи.

Таким чином, укрупнено структуру GPSS-моделі можна представити у вигляді безлічі модулів, кожен з яких описує один з протікають в досліджуваній системі паралельних процесів. Тут же наведені деякі оператори опису (команди) і виконувані оператори (блоки), які використовуються у відповідних модулях.

Оператор GPSS World, в загальному випадку, містить 4 поля:

<мітка> <операція> <операнди> ;< коментар>

Поле Мітка містить ім'я, яке може бути присвоєно оператору блоку і оператору опису.

Поле Операція містить зарезервоване слово GPSS World, що визначає функціональне призначення блоку і задає сукупність дій, які повинні бути виконані.

В поле Операнди задаються дані, необхідні для виконання операції і які становлять параметри (операнди) оператора, що розділяються комами або пробілами, наприклад: A, B, C, D або A B C D. При цьому деякі операнди є обов'язковими, тобто повинні бути завжди задані, а інші - необов'язковими, тобто можуть бути опущені при запису оператора. В останньому випадку значення цих операндів визначаються транслятором за замовчуванням.

Між двома сусідніми операндами може перебувати тільки кома або пробіл: <A, B> або <A B>. Якщо між операндами A і B після коми з'явиться пробіл: <A, B> або між ними буде знаходитися два пробіли, то це рівносильно двом комами, і операнд B сприйматиметься транслятором як третій операнд, а значення другого операнда буде визначатися за замовчуванням.

Поле Коментар розташовується після операндів, від яких відокремлюється символом «крапка з комою».

3.4.4 Процес моделювання в середовищі GPSS

Реалізація процесу моделювання полягає в переміщенні в моделі деяких рухомих об'єктів, званих транзактами. Транзакти послідовно переміщаються від блоку до блоку в заданій алгоритмом моделювання послідовності.

Транзакти створюються і знищуються в моделі за допомогою операторів (блоків): GENERATE і TERMINATE.

На початку моделювання в GPSS-моделі немає жодного транзакта. В процесі моделювання транзакти формуються в моделі в певні моменти часу відповідно до умов, заданими за допомогою блоку GENERATE. Транзакти залишають модель (знищуються), потрапляючи в блок TERMINATE. У загальному випадку, в моделі може перебувати безліч транзактів, однак в один і той же момент часу просувається тільки один транзакт. Транзакт, потрапляючи в певний блок, викликає до виконання сукупність дій, запропонованих відповідним оператором, і потім намагається увійти в наступний по порядку блок. Таке просування транзакта продовжується до тих пір, поки не відбудеться одна з наступних подій:

- Транзакт входить в блок, функцією якого є затримка транзакта на деякий заданий час (блок ADVANCE);
- Транзакт намагається увійти в блок, який "відмовляється" прийняти його до тих пір, поки в моделі не зміняться деякі умови (наприклад, блоки SEIZE, ENTER);
- Транзакт входить в блок, функцією якого є видалення транзакта з моделі (блок TERMINATE).

При виникненні одного їх перерахованих подій транзакт припиняє рух і починається переміщення в моделі іншого транзакта, тобто моделювання триває. Таким чином, моделювання полягає в переміщенні транзактів між блоками GPSS-моделі та виконанні відповідних дій.

Для зміни послідовності руху транзактів використовуються умовні та безумовні оператори, такі як TRANSFER, TEST, SELECT.

Транзакт, що просувається в моделі в даний момент часу, називається активним. Інтервал часу, протягом якого транзакт знаходиться в моделі, називається резидентним часом транзакта, а інтервал часу, протягом якого транзакт проходить від однієї довільно обраної точки моделі до іншої точки, називається транзитним часом переходу між двома цими точками.

Кожному транзакту в моделі присвоюється порядковий номер, починаючи з одиниці.

Робота реальних систем протікає в часі, для відображення якого в GPSS-моделі використовується таймер модельного часу. Зміна модельного часу відбувається шляхом його просування до найближчого події, пов'язаного зі зміною стану модельованої системи. При моделюванні СМО і ММО такими подіями є: надходження заявок в систему і завершення обслуговування заявок у вузлі ММО (обслуговуючому приладі).

Таким чином, моделювання полягає у визначенні найближчого моменту настання кожної події. Таймер модельного часу коригується автоматично відповідно до логіки, запропонованої моделлю. Таймер GPSS World може приймати будь-які значення. Одиниця часу (секунди, хвилини, години або їх частки) для таймера задається розробником моделі. Так як одиниця часу не повідомляється транслятора, то всі дані, пов'язані з часом, повинні бути виражені розробником через цю обрану одиницю.

Розглянемо більш докладно механізм зміни таймера модельного часу і логіку процесу моделювання на прикладі моделювання системи масового обслуговування з неоднорідним потоком заявок. Для формування неоднорідного потоку заявок GPSS-модель буде містити кілька операторів GENERATE за кількістю класів заявок.

На початку моделювання значення таймера модельного часу встановлюється в 0. Для всіх класів заявок, що надходять в систему, що моделюється в кожному з блоків GENERATE визначається по одному

найближчого моменту появи транзакта, що відповідає моменту надходження чергової заявки даного класу. Очевидно, що число таких моментів буде дорівнює кількості класів заявок. Серед усіх цих моментів визначається момент з найменшим значенням, тобто момент, відповідний найближчого події, і значення таймера модельного часу встановлюється рівним значенням (просувається до) цього моменту. Така зміна значення таймера модельного часу призводить до того, що відповідний транзакт з моментом надходження, рівним значенню таймера, починає рух в моделі від блоку GENERATE до наступного по порядку блоку. Рух транзакта в моделі триває до тих пір, поки він не потрапить в блок, функцією якого є затримка на деякий заданий час, або в блок, який "відмовляється" прийняти його до тих пір, поки в моделі не зміняться деякі умови. В останньому випадку транзакт залишається в попередньому блоці. Якщо транзакт входить в блок, функцією якого є видалення транзакта з моделі, то цей транзакт знищується.

Якщо в моделі є ще транзакт з таким же моментом формування, то починається його просування, причому значення таймера модельного часу не змінюється. Зміна таймера модельного часу відбувається тільки в тому випадку, якщо в моделі більше немає жодного транзакта з таким же моментом формування.

Описаний принцип просування транзактів в GPSS-моделі реалізується за допомогою так званих списків або ланцюгів (Chain). У кожен момент модельного часу все транзакти, що знаходяться в моделі, співвідносяться з одним зі списків. Залежно від приналежності транзакта того чи іншого списку, він може просуватися в моделі, бути готовим до подальшого просування, або чекати настання заданого моменту модельного часу або виконання деякого умови.

Такими списками в GPSS-моделі є:

- Список поточних подій (СПП) - містить транзакти, які можуть просуватися в моделі в поточний момент модельного часу;

- Список майбутніх подій (СМП) - містить транзакти, які очікують настання більш пізнього моменту модельного часу;

У будь-якої моделі завжди формується один список поточних і один список майбутніх подій. Решта списки формуються в міру необхідності.

У кожен момент модельного часу послідовно один за іншим просуваються тільки ті транзакти, які знаходяться в СПП, причому тільки один транзакт, який просувається в даний момент реального часу, є активним. Просування кожного транзакта здійснюється до тих пір, поки це можливо. Наприклад, якщо транзакт потрапляє в блок ADVANCE, функцією якого є затримка на деякий час, то він переводиться в СМП. Транзакт буде перебувати в СБС до тих пір, поки модельне час не стане рівним моменту, коли він може покинути блок ADVANCE. В цьому випадку транзакт буде переведений в СПП.

Транзакти, розташовані в СПП з урахуванням їх пріоритетів, вибираються послідовно один за іншим. Коли в СПП не залишається транзактів, які можуть бути просунуті в поточний момент модельного часу, відбувається зміна модельного часу, яке просувається до найближчого запланованого моменту часу для транзакта, що знаходиться першим у СМП. Цей, а також всі інші транзакти, рух яких може бути відновлено в той же момент модельного часу, переносяться з СМП в СПП, де розміщуються в порядку убування пріоритетів.

Вбудована бібліотека процедур GPSS World містить більше 20 імовірнісних розподілів, в тому числі: рівномірний (Uniform), експоненціальний (Exponential), геометричний (Geometric), Пуассона (Poisson), Бета (Beta), Гамма (Gamma), біноміальний (Binomial), дискретно-рівномірний (Discrete Uniform), трикутний (Triangular), нормальний (Normal), Парето (Pareto).

Для звернення до імовірнісного розподілу необхідно вказати ім'я бібліотечної процедури та її параметри, ув'язнені в круглі дужки і відокремлені один від одного комами:

<Ім'я процедури> (G, A, B, ...)

Тут G - номер генератора рівномірно розподілених випадкових чисел (від 1 до 999) - використовується як аргумент для формування випадкових величин із заданим законом розподілу. Інші параметри A, B, ..., кількість яких для різних розподілів становить від 1 до 4, задають безпосередньо параметри імовірнісного розподілу.

3.4.5 Оператори блоків GPSS World

У GPSS World використовуються 53 оператора блоків. Серед операторів блоків є так звані взаємодоповнюючі оператори, що представляють собою пару операторів, кожен з яких є дзеркальним відображенням іншого оператора, що означає, що сукупність дій, що реалізуються одним оператором, є протилежною по відношенню до сукупності дій, що реалізуються іншим оператором. Прикладами взаємодоповнюючих операторів можуть служити оператори GENERATE і TERMINATE, SEIZE і RELEASE, QUEUE і DEPART, ENTER і LEAVE, PREEMPT і RETURN.

Для побудови імітаційних моделей найпростіших систем і мереж масового обслуговування в середовищі GPSS World виявляється достатнім використання приблизно половини з усіх операторів блоків, які за функціональним призначенням можуть бути розбиті на наступні групи:

Оператори генерування, затримки і видалення транзактів: GENERATE, ADVANCE, TERMINATE;

- Оператори одноканальних пристроїв (приладів): SEIZE, RELEASE;
- Оператори багатоканальних пристроїв (пам'ятей): ENTER, LEAVE;
- Оператори черг: QUEUE, DEPART;
- Умовні оператори: TEST, TRANSFER, GATE;
- Оператори пріоритетного обслуговування: PRIORITY, PREEMPT, RETURN;
- Оператор логічних ключів: LOGIC;
- Інші оператори: ASSIGN, MARK, TABULATE.

Оператори можуть бути без операндів або містити від 1 до 7 операндів, деякі з яких можуть бути необов'язковими, тобто можуть бути відсутні. В останньому випадку значення необов'язкових операндів приймаються за замовчуванням. Якщо після відсутнього операнда в операторі є інші операнди, то ознакою відсутності необов'язкового операнда служить зайва кома. Наприклад, наступний запис в поле операцій: <,, C> означає, що операнди A і B не використовуються, і їх значення приймаються за замовчуванням.

3.5 Моделювання подієво-керованої вимірювальної системи в пакеті GPSS World

В даному розділі буде змодельована подієво-керована інформаційно-вимірювальна система в пакеті GPSS. В системі, що моделюється, є три датчики, два послідовно з’єднаних блоки обробки інформації та мікроконтролер, що передає дані комп’ютеру. З даних першого датчика формується сигнал події за нормальним розподілом з середнім значенням 50 мікросекунд і середньо квадратичним відхиленням 20 мікросекунд. Другий датчик формує сигнал події за нормальним розподілом з середнім значенням 200 мікросекунд і середньо квадратичним відхиленням 50 мікросекунд. Третій датчик формує сигнал події за нормальним розподілом з середнім значенням 200 мікросекунд і середньо квадратичним відхиленням 30 мікросекунд. При надходженні інформації події з першого датчику в перший блок, інформація оброблюється за нормальним розподілом з середнім значенням 4 мікросекунд і середньо квадратичним відхиленням 1 мікросекунда. При надходженні інформації події з другого датчику в перший блок, інформація оброблюється за нормальним розподілом з середнім значенням 6 мікросекунд і середньо квадратичним відхиленням 1 мікросекунда. При надходженні інформації події з третього датчику в перший блок, інформація оброблюється за нормальним розподілом з середнім значенням 6 мікросекунд і середньо квадратичним відхиленням 2 мікросекунди. При надходженні інформації події з першого

датчику в другий блок, інформація оброблюється за нормальним розподілом з середнім значенням 7 мікросекунд і середньо квадратичним відхиленням 2 мікросекунди. При надходженні інформації події з другого та третього датчику в другий блок, інформація оброблюється за нормальним розподілом з середнім значенням 7 мікросекунд і середньо квадратичним відхиленням 1 мікросекунда. Перші два блоки не мають буферу, а отже якщо на них поступає інформація події в момент обробки інформації події, що надійшла раніше, інформація події, що надходить буде втрачена. Мікроконтролер має буфер, що може зберігати 10 одиниць інформації події

3.5.1 Опис подієво-керованої вимірювальної системи в пакеті GPSS

Для імітації однієї секунди роботи системи потрібно змоделювати таймер.

```
;timer
GENERATE        1000000
TERMINATE       1
START       1
```

За базову одиницю часу приймаємо мікросекунду. В блоці START ініціалізується лічильник завершень. Процес моделювання триває до тих пір доки в лічильнику завершень додатне число. Блок TERMINATE утилізує транзакт і зменшує значення лічильника завершень на число, що вказано в першому операнді. Тому усі інші блоки TERMINATE приймають зазначення за замовчуванням – 0.

Імітація події датчиків виконується наступних операторів:

```
GENERATE        50,20
TRANSFER        ,META
```

Дані події, що надходять з датчиків представляються у вигляді транзактів, що генеруються блоком GENERATE. Після генерації транзакту відбувається безумовна передача транзакту до першого блоку обробки інформації.

```
META GATE       FV BL1,REFUSE1
```

```
        SEIZE       BL1
        ADVANCE 4,1
        RELEASE  BL1
        TRANSFER        ,META2
```

В блоці GATE перевіряється чи не зайнятий блок обробки інформації. Якщо блок не зайнятий, то вій займається оператором SEIZE та імітується обробка інформації події блоком оператором ADVANCE, що затримує транзакт на випадкову величину, що формується за нормальним законом розподілу після чого оператор RELEASE звільняє блок обробки інформації події а оператор TRANSFER передає транзакт до другого блоку обробки інформації події, що моделюються так само. Якщо блок зайнятий транзактом, що надійшов раніше то транзакт утилізується:

```
REFUSE1   TERMINATE
```

Після виходу з другого блоку транзакт потрапляє до блоку, що імітує роботу мікроконтролера:

```
METABC     TEST L      Q$BUF3,10,REFUSE3
        QUEUE     BUF3
        SEIZE       BL3
        DEPART    BUF3
        ADVANCE 33,20
        RELEASE  BL3
METTER     TERMINATE
```

Оператор TEST перевіряє чи не заповнився буфер. Якщо місця в буфері не має, то транзакт утилізується. Якщо місце є, то транзакт починає рух в моделі до наступного по порядку оператору QUEUE, який заносить транзакт (заявку) в чергу з ім'ям «BUF3». Далі транзакт продовжує рух до наступного оператору SEIZE, відповідно до якого виконує спробу зайняти одноканальний пристрій (прилад) з ім'ям «BL3». При цьому перевіряється зайнятість пристрою. Якщо прилад зайнятий обслуговуванням транзакту, що надійшов раніше, то розглянутий транзакт призупиняє свій рух і залишається в черзі до тих пір,

поки не звільниться прилад. Якщо прилад вільний, то розглянутий транзакт просувається до наступного оператору DEPART. Оператор DEPART відзначає момент покидання транзактом черги з ім'ям «BUF3» з метою збору статистики по чергах (визначається час перебування транзакта в черзі, тобто час очікування заявки). Рухаючись далі, транзакт потрапляє в оператор ADVANCE. Оператор ADVANCE затримує транзакт на випадкову величину, що формується по рівномірному закону розподілу з інтервалу 33 ± 20, моделюючи, таким чином, процес передачі даних на ПК. Подальший рух транзакта в моделі можливо тільки тоді, коли значення модельного часу досягне моменту завершення обслуговування заявки в приладі.

При попаданні транзакта в операторі RELEASE виконується сукупність дій зі звільнення приладу з ім'ям «BL3».

3.5.2 Дослідження результатів моделювання подієво-керованої вимірювальної системи

Для запуску імітації моделі необхідно запустити пункт меню Command Create simulation, після чого з’явиться журнал симуляції. Якщо симуляція пройшла успішно, з’явиться вікно звіту симуляції. Звіт симуляції вимірювальної системи:

```
      GPSS World Simulation Report - Untitled Model 2.17.1

                        Friday, June 15, 2018 01:57:00

               START TIME             END TIME   BLOCKS   FACILITIES
STORAGES
                    0.000          1000000.000      46            3
0

                    NAME                          VALUE
               BL1                                 1.000
               BL2                                 2.000
               BL3                                 3.000
               BUF1                                4.000
               BUF2                                5.000
               BUF3                                6.000
               META                               10.000
               META2                              25.000
               METABC                             35.000
```

METB	15.000
METBC2	30.000
METC	20.000
METTER	41.000
REFUSE1	42.000
REFUSE2	43.000
REFUSE3	44.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	20020	0	0
	2	PRIORITY	20020	0	0
	3	TRANSFER	20020	0	0
	4	GENERATE	4985	0	0
	5	PRIORITY	4985	0	0
	6	TRANSFER	4985	0	0
	7	GENERATE	4999	0	0
	8	PRIORITY	4999	0	0
	9	TRANSFER	4999	0	0
META	10	GATE	20020	0	0
	11	SEIZE	20020	0	0
	12	ADVANCE	20020	0	0
	13	RELEASE	20020	0	0
	14	TRANSFER	20020	0	0
METB	15	GATE	4985	0	0
	16	SEIZE	4985	0	0
	17	ADVANCE	4985	0	0
	18	RELEASE	4985	0	0

		19	TRANSFER	4985	
0	0				
	METC	20	GATE	4999	
0	0				
		21	SEIZE	4999	
0	0				
		22	ADVANCE	4999	
0	0				
		23	RELEASE	4999	
0	0				
		24	TRANSFER	4999	
0	0				
	META2	25	GATE	20020	
0	0				
		26	SEIZE	20020	
0	0				
		27	ADVANCE	20020	
0	0				
		28	RELEASE	20020	
0	0				
		29	TRANSFER	20020	
0	0				
	METBC2	30	GATE	9984	
0	0				
		31	SEIZE	9984	
0	0				
		32	ADVANCE	9984	
0	0				
		33	RELEASE	9984	
0	0				
		34	TRANSFER	9984	
0	0				
	METABC	35	TEST	30004	
0	0				
		36	QUEUE	29904	
5	0				
		37	SEIZE	29899	
0	0				
		38	DEPART	29899	
0	0				
		39	ADVANCE	29899	
1	0				
		40	RELEASE	29898	
0	0				
	METTER	41	TERMINATE	29898	
0	0				

```
    REFUSE1                  42      TERMINATE                  0
0         0
    REFUSE2                  43      TERMINATE                  0
0         0
    REFUSE3                  44      TERMINATE                100
0         0
                         45       GENERATE                     1
0         0
                         46       TERMINATE                    1
0         0

    FACILITY          ENTRIES  UTIL.    AVE. TIME AVAIL. OWNER PEND INTER RETRY
DELAY
     BL1               30004    0.140       4.662  1         0    0     0     0
0
     BL2               30004    0.210       7.004  1         0    0     0     0
0
     BL3               29899    0.981      32.798  1     30000    0     0     0
5


    QUEUE                  MAX CONT. ENTRY ENTRY(0) AVE.CONT.
AVE.TIME   AVE.(-0) RETRY
     BUF3                   10     5  29904    1431      3.294
110.165    115.702   0


    FEC XN    PRI           BDT        ASSEM   CURRENT   NEXT
PARAMETER    VALUE
     30000     2      1000005.415    30000      39      40
     30007     0      1000042.850    30007       0       1
     30006     0      1000178.898    30006       0       4
     30008     0      1000205.622    30008       0       7
     30009     0      2000000.000    30009       0      45
```

Проаналізувавши звіт, можна зробити висновок, що найбільш навантаженим блоком є мікропроцесор (UTIL = 0.981). Кількість транзактів, що поступали в момент, коли буфер був заповнений – 100. При збільшені буферу до 20 кількість втрачених транзактів зменшується до 19, до 30 – до 12, а при збільшені до 40 транзакти не втрачаються.

В даному розділі була змодельована подієво-керована вимірювальна система на пропускну здатність та були досліджені «вузькі» місця системи. Був визначений оптимальний розмір буферу.

Висновки

Одним з найважливіших етапів розробки вимірювальних систем Інтернету речей є моделювання. У розділі приведений короткий огляд методів математичного моделювання. Для моделювання подієво-керованої вимірювальної системи був використаний метод імітаційного моделювання та пакет GPSS, що дозволяє дослідити систему на пропускну здатність транзактів та оптимальний розмір буферу мікроконтролера, що дозволяє уникнути втрат транзактів.

4 РОЗРОБКА СТАРТАП ПРОЕКТУ “ОРГАНІЗАЦІЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ ТЕХНОЛОГІЙ .NET”

4.1 Опис ідеї проекту

В цьому розділі буде проведено аналіз стартап проекту. Проект передбачає розробку системи збору та аналізу результатів вимірювання температури на основі системи обробки комплексних подій та сучасного програмного забезпечення.

Враховуючи потенціал розробленої роботи, в цьому розділі розглянуто опис ідеї стартапу, та її доцільності в загальних рисах наведені в таблиці 4.1. З цієї інформації можливо зрозуміти зміст ідеї, що висувається, варіанти застосування, переваги та інновації які надаються даним стартапом.

Таблиця 4.1 Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Аналіз результатів вимірювань у реальному часі за допомогою технології обробки складних подій і відображення у веб інтерфейсі.	Вимір та демонстрація даних.	Аналіз даних у реальному часі.
	Керування результатами вимірювань.	Використання для управління.
	Аналіз оброблених даних	Швидке реагування на складні події.

Така розробка як організація систем Інтернету речей на основі технологій .NET має бути спроектована таким чином, щоб мати можливість стати бізнес проектом. Через використання сучасних технологій та доцільність таких систем існує актуальна можливість в створені рентабельної підприємницької діяльності.

Саме тому важливим розділом магістерської роботи є розробка стартап проекту. Для цього розглядаються рішення для розробки перспективного проекту з ринковою актуальністю, перспективністю, організованістю,

фінансовим аналізом, планом для просування пропозиції інвесторам, аналізом ризиків і можливостей, маркетинговим плануванням.

Проведемо аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів:

- Визначаємо перелік техніко-економічних властивостей та характеристик ідеї:

- вартість розробки;

- технології, застосовані при розробці;

- системні вимоги програмного забезпечення;

- збереження вимірювань.

Морфологічна карта стартапу наведена у таблиці 4.2.

Таблиця 4.2 – Морфологічна карта

/ п	Основні параметри	Проміжні рішення		
.	Діапазон вимірюваних температур, °С	-40…+120	-40…+80	-20…+60
.	Система комплексної обробки подій	Odysseus	WSO2 Siddhi	Esper
.	Мікроконтролер	Electric imp.	Arduino	Raspberry pi
.	Технологія розробки серверної частини	Spring	ASP.Net Core	Django
.	Брокер повідомлень	Apache ActiveMQ	Apache Kafka	Apache Qpid

Було прийнято рішення створювати систему на основі бібліотеки Esper та брокеру повідомлень Apache Kafka. Для розробки серверної частини для розробки серверної частини використовується кроссплатформний фреймворк ASP.Net Core. В якості джерела даних використовується модуль Electric imp, що складається з мікроконтролеру та датчику температури.

2. Визначаємо попереднє коло конкурентів, що вже існують на ринку, та проводимо збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку:

- Testo Saveris
- Krug2000
- Adventx

3. Проводимо порівняльний аналіз показників: для власної ідеї визначено показники, що мають:

а) гірші значення (W, слабкі);

б) аналогічні (N, нейтральні) значення;

в) кращі значення (S, сильні)

Детальний порівняльний аналіз власного проекту із проектами конкурентів наведено в табл. 4.3.

Таблиця 4.3 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Testo Saveris	Krug2000	Adventx			
1.	Вартість розробки	11000	22000	30000	6000	-	+	-

2.	Технології	Сучасні	Застарілі	Застарілі	Відносно сучасні	-	-	+
3.	Системні вимоги ПЗ	Працює на всіх ОС	Працює на Windows, Linux	Працює тільки на Windows	Працює на Windows, Mac OS	-	-	+
.	Збереження вимірів	Є можливість вибору місця збереження	Зберігає в БД та файл	Тільки в файл	Тільки в базу даних	-	-	+
.	Відомість	Відсутня	Велика	Середня	-	+	-	-

З п’яти проаналізованих характеристик, три з них є сильною стороною проекту, одна нейтральна та одна слабка. Це означає що в проекті домінують сильні сторони, що в свою чергу позитивно показує можливості даного проекту стати успішним.

4.2 Технологічний аудит проекту

В цьому розділі висвітлено технічні особливості, спектр можливих технологічних рішень. Для аналізу здійсненності проекту використаємо таблицю.

Таблиця 4.4 Технологічна здійсненність ідеї проекту

/п	Ідея проек-ту	Технології її реалізації	Наявність технологій	Доступність техно-логій

.	Розподіле на система збору, аналізу та відображення даних	Сучасні датчики вимірювання температури	+	+
		Сучасний мікроконтролер	+	+
		Мова для серверної частини C#	+	+
		База даних	+	+
		Мова розробки клієнтської частини JS	+	+
		Звичайні блокуючи підходи до розробки ПЗ	+	+
Обрана технологія реалізації ідеї проекту: система вимірювання та контролю температури з веб інтерфейсом та бекендом на ASP.NET Core				

Система комплексної обробки подій використовується для аналізу даних з великого масиву датчиків у реальному часі, та для реагування на певні події(наприклад температура перевищила певне значення). Для передання даних з датчиків до системи комплексної обробки подій використовується брокер повідомлень. Для відображення даних користувачеві через веб-інтерфейс використовується фреймворк ASP.Net Core.

Використання системи обробки складних подій для аналізу результатів вимірювань температури в реальному часі не є дуже інноваційним, але з використанням сучасних технологій у розробці програмного забезпечення стає дуже зручним і актуальним на ринку. З такою метою в даному проекті використовуються новітні технології розробки програмного забезпечення та актуальні датчики та мікроконтролери.

4.3 Аналіз ринкових можливостей запуску стартап проекту

Визначимо ринкові можливості, які можна використати під час ринкового впровадження проекту, та ринкові загрози, які можуть перешкодити його реалізації. Доцільно використати ці знання під час ринкового впровадження проекту.

Таблиця 4.5. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн/ум.од	3000000
3	Динаміка ринку (якісна оцінка)	Стабільно зростаюча
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	25%

Розглянувши таблицю стає зрозуміло, що даний ринок налічує чимало конкурентів, а ринок є зростаючим. Звичайно що головну загрозу становлять інноваційні компанії та компанії гіганти. В цих областях попит задоволено на 50-60%, це є великою можливістю увійти на цю платформу з потенціалом перехоплення суттєвої долі ринку.

Надалі визначені потенційні групи клієнтів, їх особливості. Також сформовано приблизний перелік вимог до веб-системи для кожної групи.

Таблиця 4.6 Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до послуги
1	Моніторинг температури у приміщені	Дата центри	Об’єми	Реагування на підвищення температури

2	Дистанційний контроль	Системи	Надійність	Безперебійна робота, та зручний інтерфейс

Не зважаючи на широкий спектр потреб в різних сферах, я впевнений в досягненні цих потреб даним проектом. Швидкодія та вартість є значно простішими в досягненні, а от висока точність та надійність потребує спеціальних зусиль в цьому напрямі. Але ми докладемо всіх зусиль, щоб все зробити на найвищому рівні і задовільнити всі потреби.

Таблиця 4.7 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренти	Складність запропонувати вигідніші умови	Не значне зменшення ціни
2.	Команда	Складність найняти персонал	Зручні та комфортні умови праці
3.	Застарілість технології	Поява більш інноваційних технологій	Перехід на більш новітні технології
4.	Реклама	Не правильний маркетинг.	Залучення експертів
5.	Системні збої	Перебої у передаванні вимірів на відстань	Використання дуже надійних технологій

В вище заповненій таблиці наведені фактори, які можуть в значній мірі пошкодити стартап проекту. Як протидію цим факторам запропоновані контр дії які мають покращити становище проекту в складних ситуаціях. Ці загрози не є критичними і проаналізувавши результат, проект можна вважати надійним. Але поряд із сукупністю загроз існують і певні можливості (таблиця 4.8).

Таблиця 4.8. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Швидкий притік клієнтів	Збільшення кількості клієнтів	Збільшення команди
2.	Необхідність витримувати великі навантаження	Потреба в швидкодійних системах.	Підключення великих сховищ даних
3.	Популяризація дистанційного керування	Збільшення кількості клієнтів	Зміна курсу на інший сегмент

4.	Покращення датчиків температури	Збільшення точності системи	Розробка систем з різною точністю в залежності від вимог
5.	Збільшення ASP.Net розробників	Підвищення кваліфікації робітників	Збільшення ефективності розробки.

Враховуючи перечисленні можливості, та порівнюючи їх з ризиками, можна дійти висновку що стартап перекриває ризики своїми можливостями.

При появі на ринку важливим є розуміння своїх конкурентів, їх можливостей та швидкості адаптації. Також важливо враховувати їх можливий вплив на ринок. Це детально наведено в таблиці 4.9.

Таблиця 4.9. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Монополістична конкуренція	Різноманітність товарів та послуг	Особливі пропозиції, урізноманітнення продукції
Рівень конкурентної боротьби: національний	Всі конкуренти в межах країни	Покращення точності
Внутрішньогалузева	Боротьба між економічно відокремленими товаровиробниками	Зниження витрат виробництва
Товарно-видова	Конкуренція між товарами одного виду	Просуванні власної марки, залучення спеціалістів
Нецінова конкуренція	Проводиться за рахунок модернізації	Використання новітніх технологій та сучасних датчиків і контролерів
Немарочна	Роль торгової марки незначна, хоча самі марки можуть бути присутніми на ринку	Можна нічого з цим не робити, так як роль торгової марки не значна

В цій сфері дуже велика конкуренція. Ринок налічує досить багато компаній з вже з існуючими контрактами, компаній які працюють довгий час в цій сфері. Через високий середній прибуток, дуже жорстка боротьба за клієнта. В таблиці наведені дії для поліпшення конкурентного становища компанії.

Після аналізу конкуренції нижче наведений більш детальний аналіз умов конкуренції в галузі.

Таблиця 4.10. Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Testo Saveris, Krug 2000	TechModel	Electric imp	MiroHost	Стаціонарні системи
Висновки:	Потужний конкурент.	Конкурент в майбутньому	Виробник мікросхем	Використовує систему	Системи які використовуються біля користувача

Наведені компанії є на даний момент потужними гравцями на ринку. З таблиці видно можливості для конкурентної гри для компанії.

В нижче наведеній таблиці продемонстровано фактори які забезпечать позитивні результати на конкурентному ринку.

Таблиця 4.11. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Ціна	Запропонована ціна нижче середньої
2	Час розробки	Замовників цікавить мінімальний час розробки
3	Технології	Цей проект використовує найновіші технології
4	Точність	Замовників цікавить найточніше обладнання
5.	Гнучкість	Можливість додавати функціонал до існуючої системи є важливою для клієнтів

Таблиця 4.12. Порівняльний аналіз сильних та слабких сторін «TEMPSTREAM»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з LTW systems						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	15						+	
2	Час розробки	18				+			
3	Технології	16					+		

4	Точність	15			+				
5.	Гнучкість	14					+		

З таблиць 4.10 та 4.11 бачимо, що фактори конкурентоспроможності суттєві та мають великий позитивний внесок при впровадженні нового програмного забезпечення для обробки експериментальних даних. Основною перевагою та головним досягненням є висока якість продукту, дуже висока точність, ціна.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities), пов'язаних з його здійсненням.

Таблиця 4.13. SWOT- аналіз стартап-проекту

Сильні сторони: 1.Ціна 2. Універсальність 3. Системні вимоги 4. Сучасні технології	Слабкі сторони: 1.Значний час виготовлення веб інтерфейсу 2. Слабкі можливості підтримки
Можливості: 1. Швидкий зріст кількості клієнтів 2. Необхідність витримувати великі навантаження 3. Підвищення точності 4. Збільшення ASP.Net розробників	Загрози: 1. Жорстка боротьба за клієнта 2. Досить велика кількість конкурентів 3. Малий досвід в роботі з клієнтами

Перелік ринкових загроз та ринкових можливостей складений на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. З результатів SWOT-аналізу видно, що найбільш негативний вплив на діяльність компанії по розробці програми чинить ринкове середовище.

4.4. Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Споживачами проекту обрано організації, що використовують, або можуть використовувати у своїй роботі систему контролю температури. Так як проект зосереджується на декількох сегментах, обрано стратегію диференційованого маркетингу. Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (Таблиця 4.14).

Таблиця 4.14 Базова стратегія розвитку

Обрана альтернатива розвитку проекту	**Стратегія спеціалізації** (передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок. Мета тут полягає в задоволенні потреб вибраного цільового сегменту краще, ніж конкуренти. Така стратегія може спирається на лідерство по витратах у рамках сегменту веб-система для розважальної сфери. Проте низька ринкова доля у разі невдалої реалізації стратегії може істотно підірвати конкурентоспроможність компанії.)
Стратегія охоплення ринку	Стратегія повного охоплення ринку (компанія прагне задовольнити потреби ринку в цілому. Ця стратегія може бути реалізована шляхом виготовлення сімейства універсальних веб-систем).
Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Покращення та здешевлення виробництва за рахунок масовості.
Базова стратегія розвитку	Стратегія концентрованого зростання (стратегія, яка пов'язана зі зміною продукту і (або) ринку. У разі дотримання стратегії компанія поліпшує веб-систему або починає виробляти нову, не змінюючи при цьому його призначення. Що стосується ринку, то компанія шукає можливості поліпшення свого становища на існуючому ринку або ж переходу на новий ринок).

Наступним кроком є вибір стратегії конкурентної поведінки (Таблиця 4.15).

Таблиця 4.15 □ Вибір стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Так	Ні	Стратегія наслідування лідеру

Стратегія лідерства по витратах передбачає, що компанія за рахунок чинників внутрішнього і/або зовнішнього середовища може забезпечити більшу, ніж у конкурентів маржу між собівартістю товару і середньо-ринковою ціною (або ж ціною головного конкурента). Зокрема, ця стратегія припускає, що за рахунок великих можливостей по об'ємах збуту товарів (портфеля укладених контрактів на постачання) і продуктивності підприємство може добитися менших витрат. Ця стратегія зазвичай тісно пов'язана з можливістю досягнення ефекту масштабу і досвіду. Компанії, що вибирають цю стратегію, проводять ретельний контроль за постійними витратами, знижують виробничі, збутові і рекламні витрати, проводять інвестиції, спрямовані на зменшення витрат, ретельне опрацювання конструкції нових товарів. [1]

Переваги стратегії за Ж.-Ж. Ламбеном [1] :

- фірма здатна протистояти своїм прямим конкурентам навіть у разі цінової війни і в змозі отримувати прибуток при ціні, мінімально допустимій для конкурентів;
- сильні клієнти не можуть добитися зниження ціни нижче рівня, прийнятного для найбільш сильного конкурента;
- низькі витрати забезпечують захист проти сильних постачальників, оскільки дають фірмі велику гнучкість у разі підвищення вхідних витрат;
- низькі витрати створюють бар'єр входу для нових конкурентів і одночасно хороший захист проти товарів-замінників.

В ході конкурентної боротьби з використанням цієї стратегії з ринку вимушені будуть піти фірми, менш ефективні з точки зору величини і структури витрат, нездібні до проведення технологічних новацій, спрямованих на зниження витрат.

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Якість	Розширення первинного попиту	Висока якість продукту. Швидкість, гнучкість	Швидкість, гнучкість, якість
2	Обслуговування	Розширення первинного попиту	Відносно швидке обслуговування та гнучке встановлення	Легкість, довгострокова гарантія
3	Ціна	Наступальна	Доступна цінова політика	Гнучкість, доступність

Таблиця 4.16 Визначення стратегії позиціонування

Основними вимогами до товару цільової аудиторії є Якість, ціна, обслуговування. Було обрано асоціації на базі вимог цільової аудиторії, які мають сформувати комплексну позицію власного проекту – швидкість, модульність та доступність.

4.5 Розробка маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 4.17 підсумуємо результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.17 Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
Розподілена система вимірювання температури	Менша ціна в порівнянні з конкурентами та зручний інтерфейс	Універсальність застосування, сучасна апаратна частина та програмне забезпечення

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціювання, визначену специфіку поведінки клієнтів.

Таблиця 4.18 Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	1. Розподілена система вимірювання температури з використанням системи обробки складних подій.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1.Довговічність (немає строку давності)	Нм	Вр
	2. Швидкодія	М	Тх
	3. Модель дефектів	М	Тл
	4. Блокування некоректних дій	М	Тх
	5. Умови експлуатації	Нм	Ор
	Якість: ГОСТ, ISO		
	Пакування: мікропроцесорна система вимірювання відстані та пакет програмного забезпечення		
	Марка: поки що відсутня		
III. Товар із підкріпленням	До продажу: потребує навичок, які забезпечує технічна підтримка		
	Після продажу: підтримка клієнтів		
За рахунок чого програмне забезпечення буде захищено: для користування ним для кожного користувача буде видаватися обліковий запис			

Опис трьох рівнів моделі товару показав, що основний задум даного стартап проекту полягає у створенні розподіленої системи вимірювання температури. Його перевагою є точність і використання сучасних технологій. До продажу і після продажу на них очікує технічна підтримка. Проект буде захищено шляхом ліцензії.

Наступним кроком є визначення цінових меж, якими необхідно керуватися при встановленні ціни на потенційний товар, це передбачає аналіз цін товарів конкурентів, та доходів споживачів продукту (табл. 4.19).

Таблиця 4.19 Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу

1	-	1800-2500 $	20000 – 40000 грн	1500 – 2500 $

Виконавши аналіз рівня цін на товари-замінники, рівень цін на товари-аналоги та рівень доходів цільової групи споживачів було сформовано нижню 1500 $ та верхню 2500 $ межі встановлення ціни на товар, що дає цінову перевагу перед товарами конкурентів.

Таблиця 4.20 Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Замовлення з підписанням договору купівля-продаж	Встановлення контактів зі споживачами, їх підтримка; Чітка доставка в строки, та без помилок в розрахунках.	Виробник безпосередньо продає товар клієнту	Прямий продаж безпосередньо від виробника; непрямий - через системи типу «інтернет»; від виробника на замовлення підприємств

Зазначені функції збуту, глибина каналу формують оптимальну систему збуту, яка відбуватиметься через сайт розробника, прямий продаж безпосередньо від виробника, від виробника на замовлення підприємств. Завдяки цьому можна легко встановлювати контакти зі споживачами та їх подальшу підтримку, організовувати дослідницьку роботу зі збору маркетингової інформації та розробку і реалізацію програм підтримки лояльності клієнтів.

Таблиця 4.21 Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Покупець організація-споживач оцінює характеристики, технічну компетентність та здатність забезпечити умови постачання	Виставки, інтернет	Технічні характеристики, якість, зовнішній вигляд, сучасні технології	Поширення характеристик продукту	Демонстрація переваг нашого продукту перед існуючими аналогами

Зважаючи на те, що цільові клієнти більшу частину інформації про нові програми отримують через тематичні виставки, мережу інтернет, то доцільними ключовими позиціями було обрано: сучасні технології, технічні характеристики системи. Завданням рекламного повідомлення є зацікавлення та позиціювання товару новим клієнтам.

4.6 Висновки

Узагальнюючи проведений аналіз стартап проекту можна зробити висновок, що на даний момент попит на систему є. Це можна підтвердити позитивною динамікою ринку та потребую потенційних клієнтів у використані схожих систем.

Конкуренція на ринку України в цій області є значною, але ми маємо багато переваг для легкого входу на український ринок. За кордоном вже існують компанії, що працюють в сфері аналізу даних у реальному часі, проте наявні фактори конкурентоспроможності робить можливим і вихід на закордонний ринок.

Цільовою аудиторією є різні компанії які працюють з дата центрами. Найкраще з усіх альтернатив було досліджено, після чого було прийнято рішення максимально збільшити точність вимірів та зробити зручний

інтерфейс. Також ця система може бути використана як пожежна сигналізація, для цього в систему можна додати датчики задимленості та аналізувати дані з них разом з даними, що надходять з датчиків температури.

До продажу і після продажу на них очікує технічна підтримка. Програмне забезпечення буде захищено шляхом ліцензії.

Зважаючи на те, що цільові клієнти більшу частину інформації про нові програми отримують через тематичні виставки, мережу інтернет, то доцільними ключовими позиціями було обрано: сучасні технології, технічні характеристики системи. Завданням рекламного повідомлення є зацікавлення та позиціювання товару новим клієнтам.

Все більше компаній переходять на автоматизовані системи замірів з дистанційним керування, які створенні на основі сучасних датчиків та мають зручний інтерфейс написаний на сучасних фреймворках. Здійснення запропонованого проекту є доцільним, оскільки технології та проекти в IT-сфері розвиваються і ще більше будуть популярні у майбутньому.

ВИСНОВКИ

IoT - це багатообіцяюча технологія, яка може змінити спосіб життя. Оскільки вона є відносно новою, вона стикається з багатьма проблемами. При реалізації системи IoT досягнення масштабованості є однією з ключових проблем, яку потрібно вирішити. Можливо досягти масштабованості, якщо інтегрувати рішення IoT та хмарні обчислення. Хмарні технології надають практично необмежений об'єм зберігання та обчислювальних ресурсів, і це ідеально підходить для IoT-додатків.

В роботі надано докладний опис особливостей платформи Electric Imp, що орієнтована на хмарні IoT середовища, описані можливості та функції платформи. Було проведено детальний огляд комплекту impExplorer та Imp001. Досліджено два режими роботи операційної системи impOS. Розглянуто головні етапи в налаштуванні проекту в хмарному середовищі impCentral: реєстрація в impCentral, активація імпу за допомогою BlinkUp технології та створення продукту та пристрою в даному середовищі. Описано розробку коду для хмарного агенту та пристрою в середовищі розробки impCentral за допомогою мови Squirrel.

Більшість сценаріїв реального життя (такі як розроблений в роботі сценарій виявлення пожежі) потребують дій у реальному часі. Складна обробка подій допомагає обробляти величезну кількість подій «на льоту». По мірі виникнення подій можливо отримувати інформацію з подій у режимі реального часу. Для аналізу подій в роботі використовуються методи CEP пакету NEsper для оточення .NET.

Важливим етапів проектування систем IoT є їх моделювання. У магістерській дисертації наведено огляд методів математичного моделювання. Для моделювання системи IoT був використаний метод імітаційного моделювання та пакет GPSS, що дозволяє дослідити систему на пропускну здатність транзактів (подій) та визначити оптимальний розмір буферу мікроконтролерної системи, що дозволяє уникнути втрат транзактів (подій).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Event-driven architecture [Electronic resource]. https://en.wikipedia.org/wiki/Event-driven_architecture. – Last access: 26.08.2019.

[2] D Robins. Complex event processing. In Second International Workshop on Education Technology and Computer Science. Wuhan. Citeseer, 2010.

[3] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR), 44(3):15, 2012.

[4] Esper Reference [Electronic resource]. – Mode of access: http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html. – Last access: 26.09.2019.

[5] Electric Imp Developer Guide [Electronic resource]. – Mode of access: https://developer.electricimp.com/hardware/resources/reference-designs/explorerkit. – Last access: 20.02.2018.

[6] Salesforce REST API Guide [Electronic resource]. – Mode of access: https://developer.salesforce.com/page/REST_API. – Last access: 26.08.2019.

[7] Squirrel Documentation [Electronic resource]. – Mode of access: http://squirrel-lang.org/squirreldoc/. – Last access: 26.09.2019.

[8] Рябов О.А. Моделирование процессов и систем. Красноярск 2008 – 130с.

[9] Можаев А.С. Общий логико-вероятностный метод анализа надежности сложных систем. Ленинград 1988 – 68с.

[10] Бобков С.П. Моделирование систем учебное пособие Иваново 2008 – 157с.

[11] Алиев Т.И. Основы моделирования дискретных систем учебное пособие Санкт-Петербург 2009 – 363с.

[12] Лузина Л.И. Компьютерное моделирование учебное пособие Томск – 2001 – 105с.

[13] Кудрявцев Е.М. GPSS World Основы имитационного моделирования различных систем ДМК Пресс, 2004 – 318с.

[14] Томашевский В., Жданова Е. Имитационное моделирование в среде GPSS. – М.:Бестселлер, 2003. – 416 с.

ДОДАТОК А. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

TemperatureEvent.cs

```
using System;

namespace EventModels
{
 public class TemperatureEvent
 {
      public decimal Value { get; set; }

      public int SensorId { get; set; }

      public DateTime Time { get; set; }
 }
}
```

Room.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace FireAlarmConsole.Model
{
 public class Room
 {
      public int Id { get; set; }

      private List<int> _sensorIds = new List<int>();

      public int Number { get; set; }

      public int Floor { get; set; }

      public IEnumerable<int> SensorIds => _sensorIds;

      public void AddSensorId(int id) => _sensorIds.Add(id);
 }
}
```

Program.cs

```
using com.espertech.esper.client;
using FireAlarmConsole.Model;
using System;
using System.Collections.Generic;
using EventModels;
using com.espertech.esper.compat;
using System.Threading;
```

```
    using System.Linq;
    using System.Globalization;

    namespace FireAlarmConsole
    {
     class Program
     {
          static void Main(string[] args)
          {
               CultureInfo.CurrentCulture =
CultureInfo.InvariantCulture;

               var sensorIdConter = 1;
               var rooms = new List<Room>();

               for (int floor = 0; floor < 10; floor++)
               {
                    for (int number = 0; number < 10; number++)
                    {
                         var room = new Room
                         {
                              Id = 10 * floor + number + 1,
                              Floor = floor + 1,
                              Number = 10 * floor + number + 1
                         };
                         for (int sensor = 0; sensor < 5; sensor++)
                         {
                              room.AddSensorId(sensorIdConter++);
                         }

                         rooms.Add(room);
                    }
               }

               var engine =
EPServiceProviderManager.GetDefaultProvider();
```

```
     engine.EPAdministrator.Configuration.AddEventType<TemperatureE
vent>();

     engine.EPAdministrator.Configuration.AddEventType<Room>();

     engine.EPAdministrator.Configuration.AddEventType("RoomToSenso
r", new Dictionary<string, object>()
               {
                    ["SensorId"] = "int",
                    ["RoomId"] = "int"
               });

               var avaregeTemperatureStatemant = $@"
                    select
                         avg(Value) as roomTemperature,
                         R.Number as roomNumber,
                         R.Floor as roomFloor
                    from TemperatureEvent#time(30 sec)
                         join  RoomToSensor#length({rooms.Sum(x  =>
x.SensorIds.Count())})  as  RTS  on  TemperatureEvent.SensorId  =
RTS.SensorId
                         join  Room#length({rooms.Count})  as  R  on
RTS.RoomId = R.Id
                    Group by R.Number, R.Floor
                    order by R.Number";
               var                       stat                      =
engine.EPAdministrator.CreateEPL(avaregeTemperatureStatemant);

               var fireStatemant = $@"
                    select
                         R.Number as roomNumber,
                         R.Floor as roomFloor
                    from TemperatureEvent#time(30 sec)
```

```
                    join RoomToSensor#length({rooms.Sum(x =>
x.SensorIds.Count())}) as RTS on TemperatureEvent.SensorId =
RTS.SensorId
                    join Room#length({rooms.Count}) as R on
RTS.RoomId = R.Id
                Group by R.Number
                Having avg(Value) > 49.95";
            var                    statFire                    =
engine.EPAdministrator.CreateEPL(fireStatemant);

            foreach (var room in rooms)
            {
                engine.EPRuntime.SendEvent(room);
                foreach (var sensorId in room.SensorIds)
                {
                    engine.EPRuntime.SendEvent(new
Dictionary<string, object> { ["SensorId"] = sensorId, ["RoomId"] =
room.Id }, "RoomToSensor");
                }
            }

            var locker = new object();

            var sender = new Thread(() => {
                var rand = new Random((int)DateTime.Now.Ticks);
                while (true)
                {
                    lock (locker)
                    {
                        engine.EPRuntime.SendEvent(new
TemperatureEvent() { SensorId = rand.Next(sensorIdConter), Time =
DateTime.Now, Value = (decimal)(50 * rand.NextDouble()) });
                    }
                    Thread.Sleep(1);
                }
            });
```

```
            sender.Start();

            var alarmed = false;

            statFire.AddEventHandlerWithReplay((s, e) => {
                if (e.NewEvents is null)
                {
                    return;
                }
                Console.Clear();
                Console.ForegroundColor = ConsoleColor.Red;
                alarmed = true;

                foreach (var res in e.NewEvents)
                {
                    Console.WriteLine("FIRE in room {0} on {1}
floor", res.Get("roomNumber"), res.Get("roomFloor"));
                }
            });

            while (!alarmed)
            {
                Console.Clear();
                lock (locker)
                {
                    foreach (var res in stat)
                    {
                        Console.WriteLine("avarage
temperature {0:N2} in room {1} on {2} floor",
res.Get("roomTemperature"), res.Get("roomNumber"),
res.Get("roomFloor"));
                    }
                }
                Thread.Sleep(1000);
            }
```

```
        }
     }
    }

    // Log the URLs we need
    server.log("Get Current Temperature: " + http.agenturl() +
"?temp");

    Gresponse <- null;

    function requestHandler(request, response) {
      try {
        // Check if the user sent led as a query parameter
        //if ("temp" in request.query) {
        //    device.send("get.temperature");
        //  }
        ::Gresponse = response;
        device.send("get.temperature", null);

      } catch (ex) {
        response.send(500, "Internal Server Error: " + ex);
      }
    }

    function motion(data) {
      local message = format("%f", data);
      ::Gresponse.send(200, message);
      server.log(message);
    }

    #require "HTS221.device.lib.nut:2.0.1"
    #require "LPS22HB.device.lib.nut:2.0.0"
    #require "WS2812.class.nut:3.0.0"

    // Define constants
```

```
const SLEEP_TIME = 1;

// Declare Global Variables
tempSensor <- null;
pressureSensor <- null;
led <- null

// Define functions
function takeReading(ttr){
  local reading = tempSensor.read();

  // Send 'conditions' to the agent
  agent.send("reading.sent", reading.temperature);

  led.set(128, [0,128,128]).draw();
   imp.wakeup(0.1, shutDownLed);
}

function shutDownLed() {
  led.set(0, [0,0,0]).draw();
}

// Start of program

// Configure I2C bus for sensors
local i2c = hardware.i2c89;
i2c.configure(CLOCK_SPEED_400_KHZ);

tempSensor = HTS221(i2c);
tempSensor.setMode(HTS221_MODE.ONE_SHOT);

pressureSensor = LPS22HB(i2c);
pressureSensor.softReset();

// Configure SPI bus and powergate pin for RGB LED
local spi = hardware.spi257;
```

```
    spi.configure(MSB_FIRST, 7500);
    hardware.pin1.configure(DIGITAL_OUT, 1);
    led <- WS2812(spi, 1);

    agent.on("get.temperature", takeReading);

    using System;
    using System.Net;
    using System.IO;
    using com.espertech.esper.client;
    using System.Threading;
    using System.Linq;


    namespace ConsoleApp2
    {
        class Program
     {
          static void Main(string[] args)
          {
               var                    engine                    =
EPServiceProviderManager.GetDefaultProvider();

engine.EPAdministrator.Configuration.AddEventType<TempEvent>();
               var  eQuery  =  "select  avg(Temperature)  avgTemp,
count(Temperature) measurementCount from TempEvent#time(30 sec)";
               var                    stat                    =
engine.EPAdministrator.CreateEPL(eQuery);

               new Thread((o) => {
                   while (true)
                   {
                       var                wr                =
WebRequest.Create(@"https://agent.electricimp.com/FfJr1VrZ9MMN");
                       engine.EPRuntime.SendEvent(new  TempEvent()
{
```

```
                            Temperature     =     double.Parse(new
StreamReader(wr.GetResponse().GetResponseStream()).ReadLine()
                            ,
System.Globalization.NumberFormatInfo.InvariantInfo)
                        });
                    }
                }).Start();

                Thread.Sleep(10000);

                while (true)
                {
                    var res = stat.First();
                    Console.WriteLine("avarage   temperature   {0}
with         {1}         measurements",         res.Get("avgTemp"),
res.Get("measurementCount"));
                    Thread.Sleep(1000);
                }
            }
     }

        public class TempEvent
        {
            public double Temperature { get; set; }
        }
    }
```

```
         BL1   EQU  1
         BL2   EQU  2
         BL3   EQU  3
         BUF1  EQU  4
```

```
BUF2  EQU  5
BUF3  EQU  6

 GENERATE  50,20
 PRIORITY  1
 TRANSFER  ,META

 GENERATE  200,50
 PRIORITY  2
 TRANSFER  ,METB

 GENERATE  200,30
 PRIORITY  2
 TRANSFER  ,METC

META  GATE FV BL1,REFUSE1
 SEIZE     BL1
 ADVANCE   4,1
 RELEASE   BL1
 TRANSFER  ,META2

METB  GATE FV BL1,REFUSE1
 SEIZE     BL1
 ADVANCE   6,1
 RELEASE   BL1
 TRANSFER  ,METBC2

METC  GATE FV BL1,REFUSE1
 SEIZE     BL1
```

```
 ADVANCE   6,2
 RELEASE   BL1
 TRANSFER  ,METBC2

META2 GATE FV BL1,REFUSE2
 SEIZE     BL2
 ADVANCE   7,2
 RELEASE   BL2
 TRANSFER  ,METABC

METBC2     GATE FV BL1,REFUSE2
 SEIZE     BL2
 ADVANCE   7,1
 RELEASE   BL2
 TRANSFER  ,METABC

METABC     TEST L    Q$BUF3,40,REFUSE3
 QUEUE     BUF3
 SEIZE     BL3
 DEPART    BUF3
 ADVANCE   33,20
 RELEASE   BL3
METTER     TERMINATE

REFUSE1    TERMINATE
REFUSE2    TERMINATE
REFUSE3    TERMINATE

;timer
```

```
GENERATE  1000000

TERMINATE 1

START     1
```

ДОДАТОК Б. СПИСОК ПУБЛІКАЦІЙ

Баштовий М.Ю., Богомазов С.А. ОРГАНІЗАЦІЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ ТЕХНОЛОГІЙ .NET // Збірник тез доповідей XVIII Міжнародної науково-технічної конференції ПРИЛАДОБУДУВАННЯ: стан і перспективи, 15-16 травня 2019 року, Київ, ПБФ КПІ ім. Ігоря Сікорського, 2019, с.203-204.

УДК 621.3.087.44

ОРГАНІЗАЦІЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ ТЕХНОЛОГІЙ .NET

Баштовий М.Ю., Богомазов С.А.
КПІ ім. Ігоря Сікорського
E-mail: maksimbastovoj@gmail.com

Класичні веб-додатки приймають дані від користувачів та зберігають їх на протязі невизначеного часу. Але додатки, що отримують безперервні потоки даних та обробляють їх у реальному часі не можуть використовувати традиційні бази даних, в яких для запису та зчитування даних витрачається багато часу. Прикладом таких додатків є додатки Інтернету речей.

Розроблено демонстраційну систему Інтернету речей, в якій інформація про температуру дистанційно отримується за допомогою одноплатного комп’ютера Raspberry Pi. До нього через інтерфейс GPIO під’єднаний датчик температури. Прилад запрограмований на щосекундну відправку даних показів датчику до веб-сервісу, що діє як концентратор для збору та обробки даних від приладу [1]. Для обробки даних веб-сервіс використовує бібліотеку NEsper, що дозволяє обробляти дані без запису їх на диск.

NEsper – це .NET бібліотека для складної обробки подій. Правила обробки подій описуються за допомогою SQL-подібної мови EPL (Event Processing Language), однак сама логіка роботи NEsper принципово відрізняється від запитів до баз даних. Замість того щоб зберігати події і періодично виконувати запити, NEsper зберігає правила (запити) і пропускає крізь них потік подій. NEsper дозволяє швидко розробляти додатки, що обробляють великий об’єм вхідних повідомлень та подій [2]. Це дозволяє, наприклад, розраховувати

середнє значення температури в рухомому часовому вікні для постійного аналізу та повідомлення про критичну ситуацію. Для візуалізації результатів в розробленій системі використовується веб-додаток, розроблений з використанням фреймворку ASP.NET MVC [3]. Для передачі даних між сервером та клієнтом результати на сервері серіалізуються в XML-вигляд, а на клієнті – десеріалізуються.

Використання в додатках Інтернету речей можливостей бібліотеки NEsper дозволило забезпечити поточну обробку та аналіз даних без запису на диск та зменшити часові затримки при обробці даних в системах реального часу, що керуються подіями.

Ключові слова: Інтернет речей, обробка даних.

Література

[1] https://msdn.microsoft.com/ru-ru/magazine/hh852591.aspx
[2] http://www.espertech.com/esper
[3] *Адам Фримен.* ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов, 5-е издание.— М.: «Вильямс», 2014. — 736 с.

2. Баштовий М.Ю., Зінченко В.П ПРОГРАМНІ СИСТЕМИ МОДЕЛЮВАННЯ ІОС. GPSS ТА ІН.// Тези доповідей учасниківXXIнауково-технічної конференціїстудентів та молодих учених. ГІРОТЕХНОЛОГІЇ, НАВІГАЦІЯ, КЕРУВАННЯРУХОМ ТА КОНСТРУЮВАННЯАВІАЦІЙНО-КОСМІЧНОЇ ТЕХНІКИ, 21березня2018року, Київ, «Політехніка», 2018, с 9 - 10

УДК 519.6

Баштовий М.Ю., Зінченко В.П

ПРОГРАМНІ СИСТЕМИ МОДЕЛЮВАННЯ ІОС. GPSS ТА ІН

Актуальність. Програмні системи моделювання є корисним інструментом для математичного моделювання природніх систем у фізиці, астрофізиці, хімії, біології, та інженерії. Маніпулюючи моделлю, можна отримати нові знання про об'єкт чи для наближеної оцінки поведінки систем, занадто складних для аналітичного дослідження.

З розробкою швидкодіючих ЕОМ використання моделювання значно зросли. Представлення системи математичною моделлю, перетворення моделі в команди для ЕОМ та виконання програми на ЕОМ дали можливість моделювати великі та складні системи, що дозволило моделювати на ЕОМ самі ЕОМ. Таким чином ЕОМ беруть участь в моделюванні в двох ролях: обчислювальні засоби та об'єкт моделювання.

Нові наукові та технічні результати. Система складається із окремих взаємодіючих компонентів. Кожна компонента може бути системою та мати свій стан. Дії однієї компоненти системи можуть виконуватись одночасно з діями інших компонент. Паралельна природа дій в системі створює труднощі при моделюванні. Оскільки компоненти взаємодіють, необхідно встановити синхронізацію.

Для моделювання систем, що містять взаємодіючі паралельні компоненти необхідно використовувати мережі Петрі. В загальноприйнятому підході застосування мережі Петрі в проектуванні є необхідним постійне перетворення системи в модель у вигляді мережі Петрі. Можна запропонувати інший, більш радикальний підхід в якому весь процес проектування та визначення характеристик приводиться в термінах мережей Петрі. Методи аналізу застосовуються тільки для створення проекту мережі Петрі, вільного від помилок. Тут задача полягає в перетворенні мережі Петрі в реальну робочу систему.

Ці два підходи використання мережей Петрі в процесі проектування пропонують досліднику мережей Петрі задачі різного типу. В першому випадку необхідна розробка методів моделювання систем мережами Петрі, а в другому випадку мають бути розроблені методи реалізації мережей Петрі для визначення властивостей системи.

Практична застосовність. Мережі Петрі – інструмент моделювання та дослідження систем. Теорія мереж Петрі дає можливість моделювати системи математичним представленням у вигляді мережі Петрі.

Аналіз мереж Петрі допоможе отримати важливу інформацію про структуру та динамічну поведінку системи. Ця інформація буде корисною для оцінки системи та створення пропозицій щодо її вдосконаленню.
[4]